Introduction to Artificial Intelligence

Daniel RACOCEANU

SUPPORT VECTOR MACHINES

Update Oct. 2020

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS
DEPUIS 1257

Support Vector Machine (SVM)

- A classifier derived from statistical learning theory by Vapnik, et al. in 1992
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task
- Currently, SVM is widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.
- Also used for regression

Vladimir Vapnik

Outline

- Linear Discriminant Function
- Large Margin Linear Classifier
- Nonlinear SVM: The Kernel Trick
- Demo of SVM

Discriminant Function

- A classifier *g* is said to assign a feature vector *x* to class ω_i if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \qquad \text{for all} \quad j \neq i$$

- For two-category case, $g(\mathbf{x}) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x})$

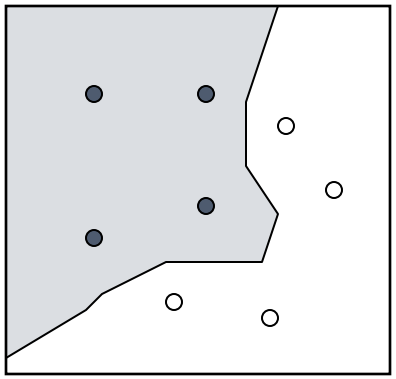
$$\text{Decide } \omega_1 \text{ if } g(\mathbf{x}) > 0; \text{ otherwise decide } \omega_2$$

- Minimum-Error-Rate Classifier

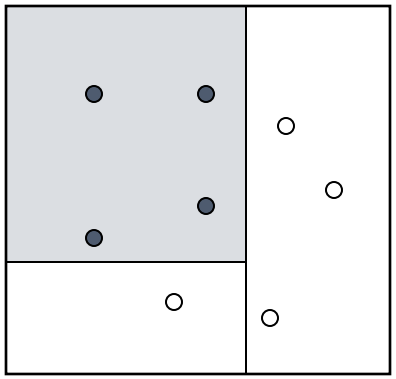
$$g(\mathbf{x}) \equiv p(\omega_1 \mid \mathbf{x}) - p(\omega_2 \mid \mathbf{x})$$

Discriminant Function

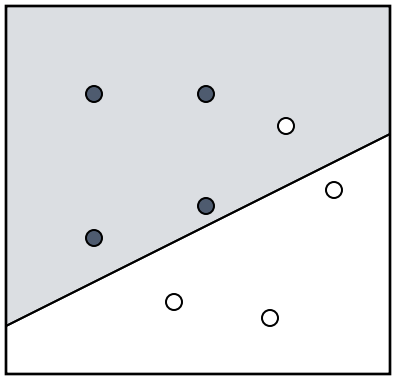
- It can be an arbitrary function of *x* , such as:



Nearest Neighbor

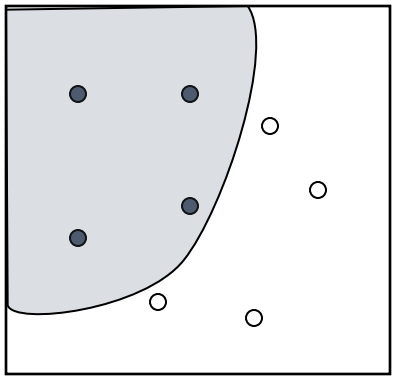


Decision Tree



Linear Functions

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$



Nonlinear Functions

Linear Discriminant Function

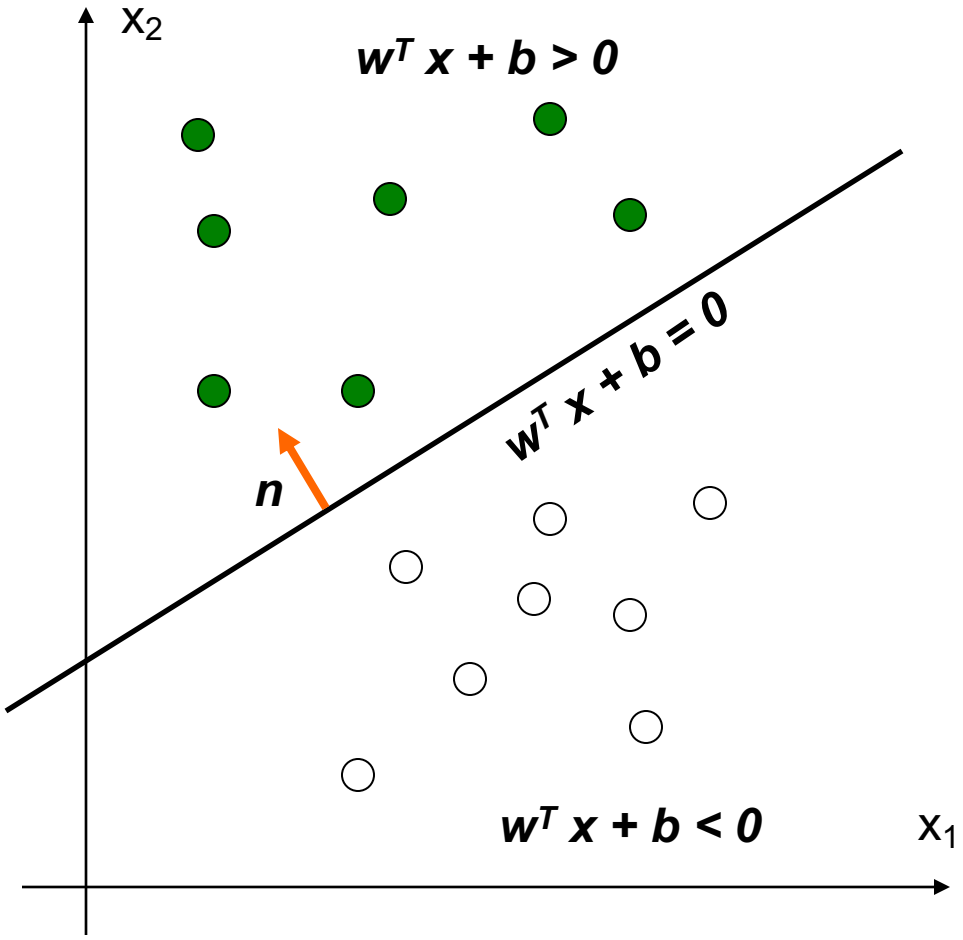
- g(x) is a linear function:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- A hyper-plane in the feature space

- (Unit-length) normal vector of the hyper-plane:

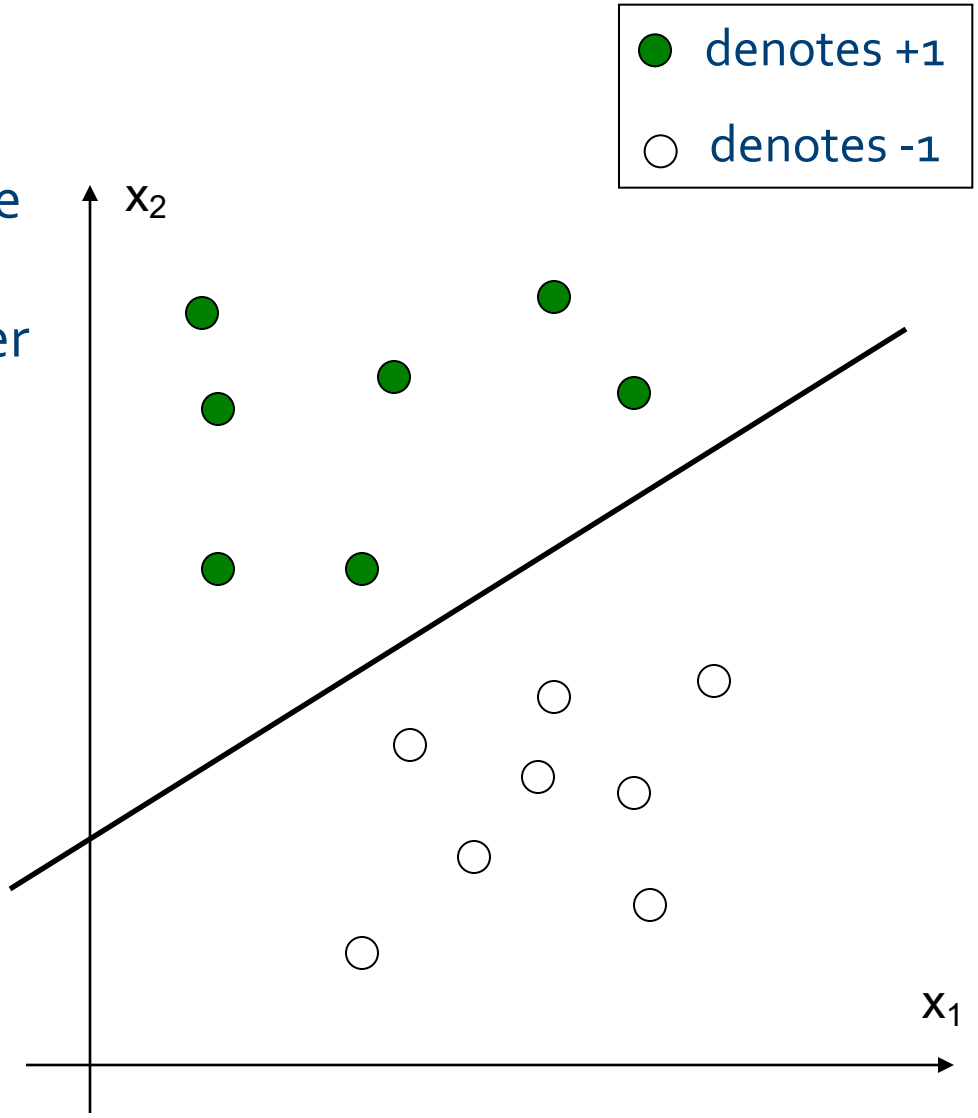
$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

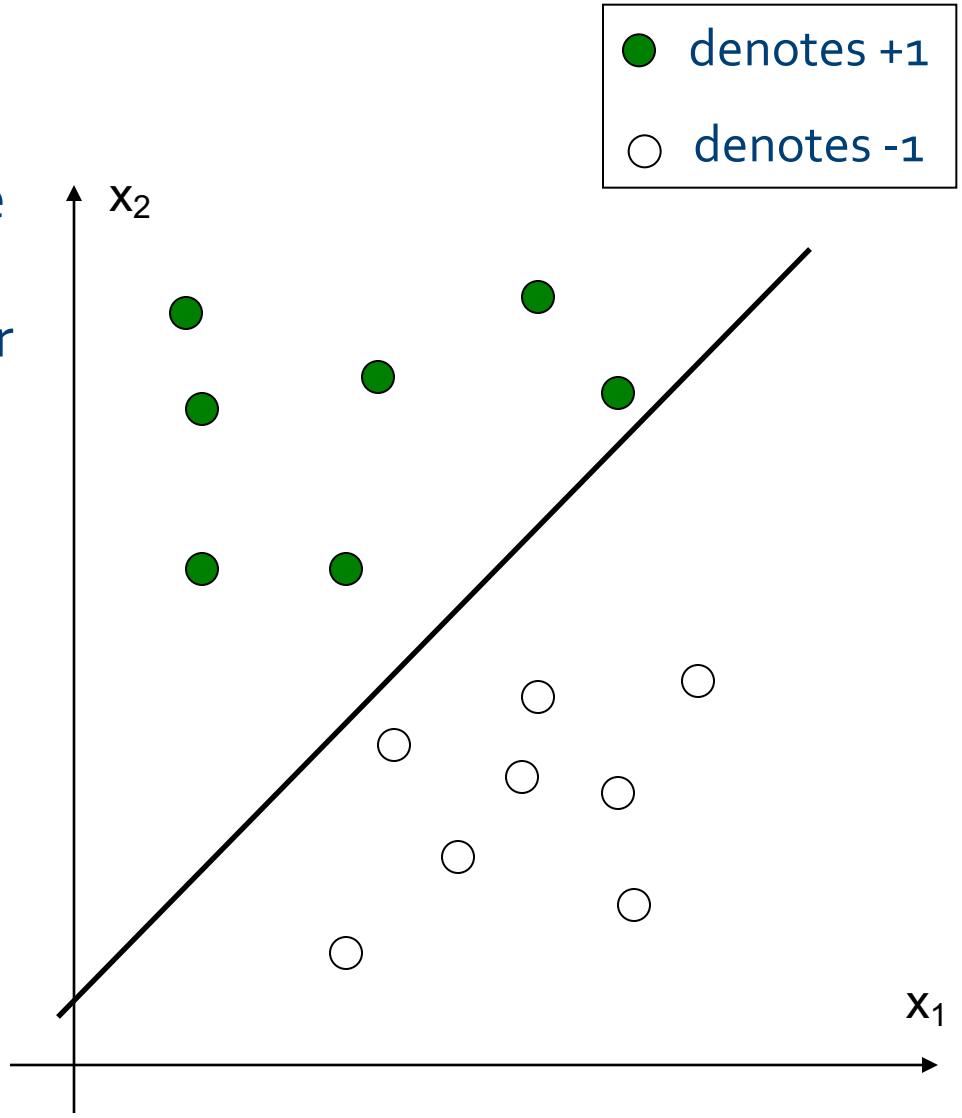
- Infinite number of possible answers!



Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

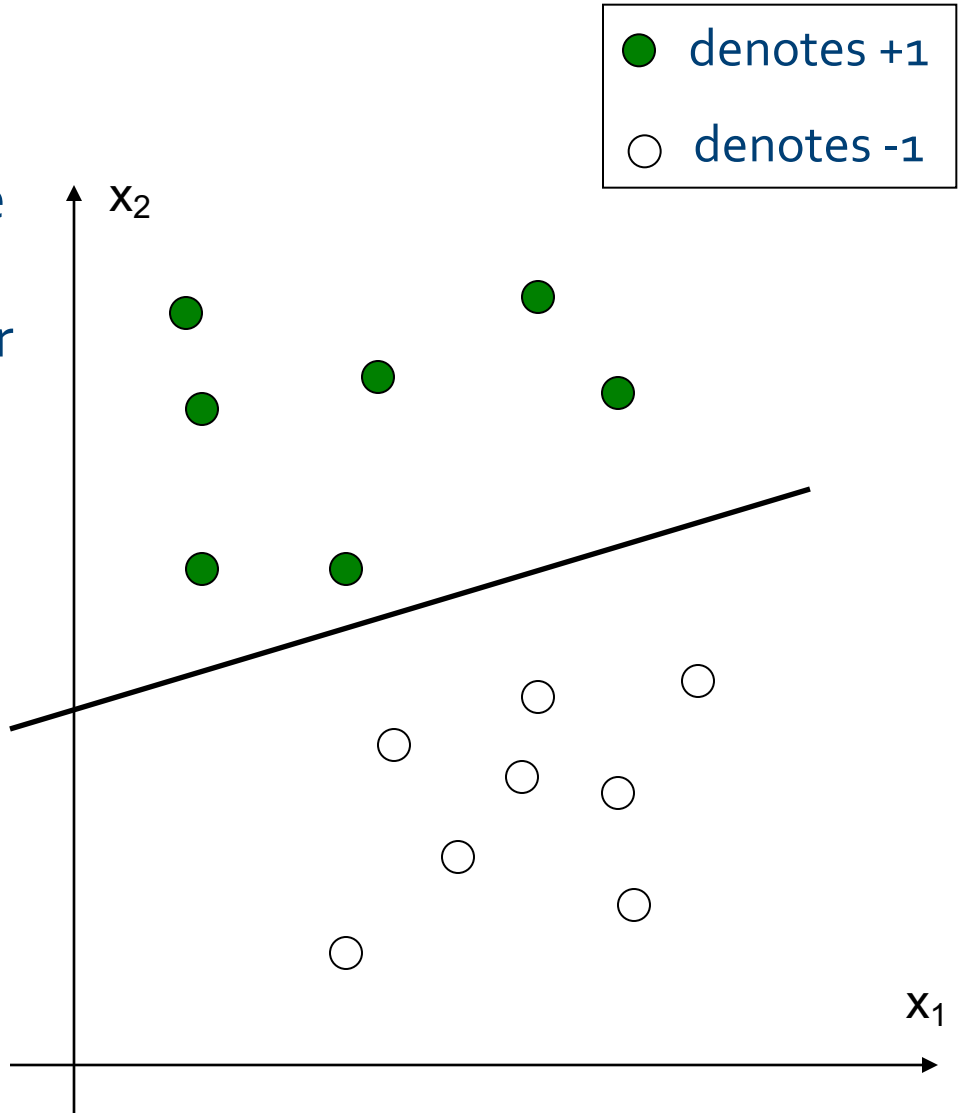
- Infinite number of possible answers!



Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of possible answers!

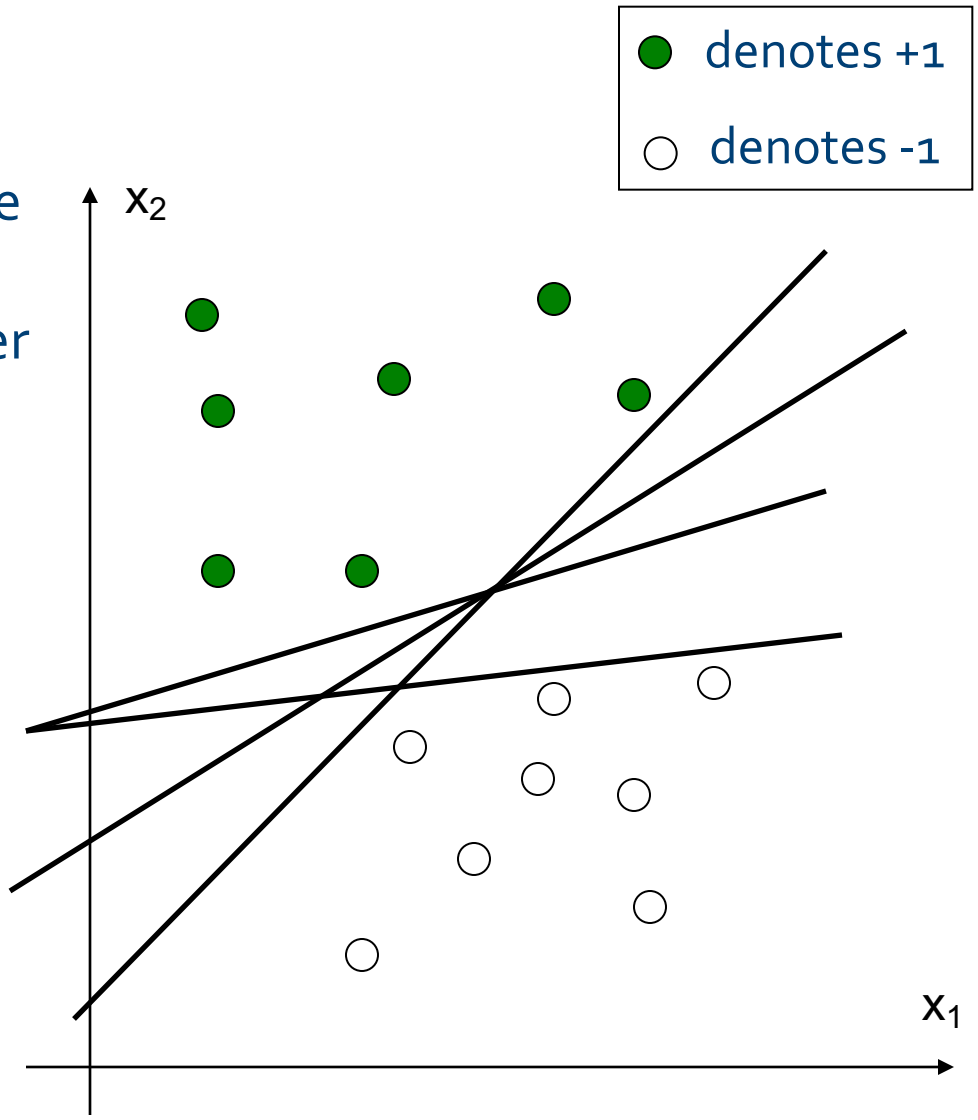


Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of possible answers!

- Which one is the best?

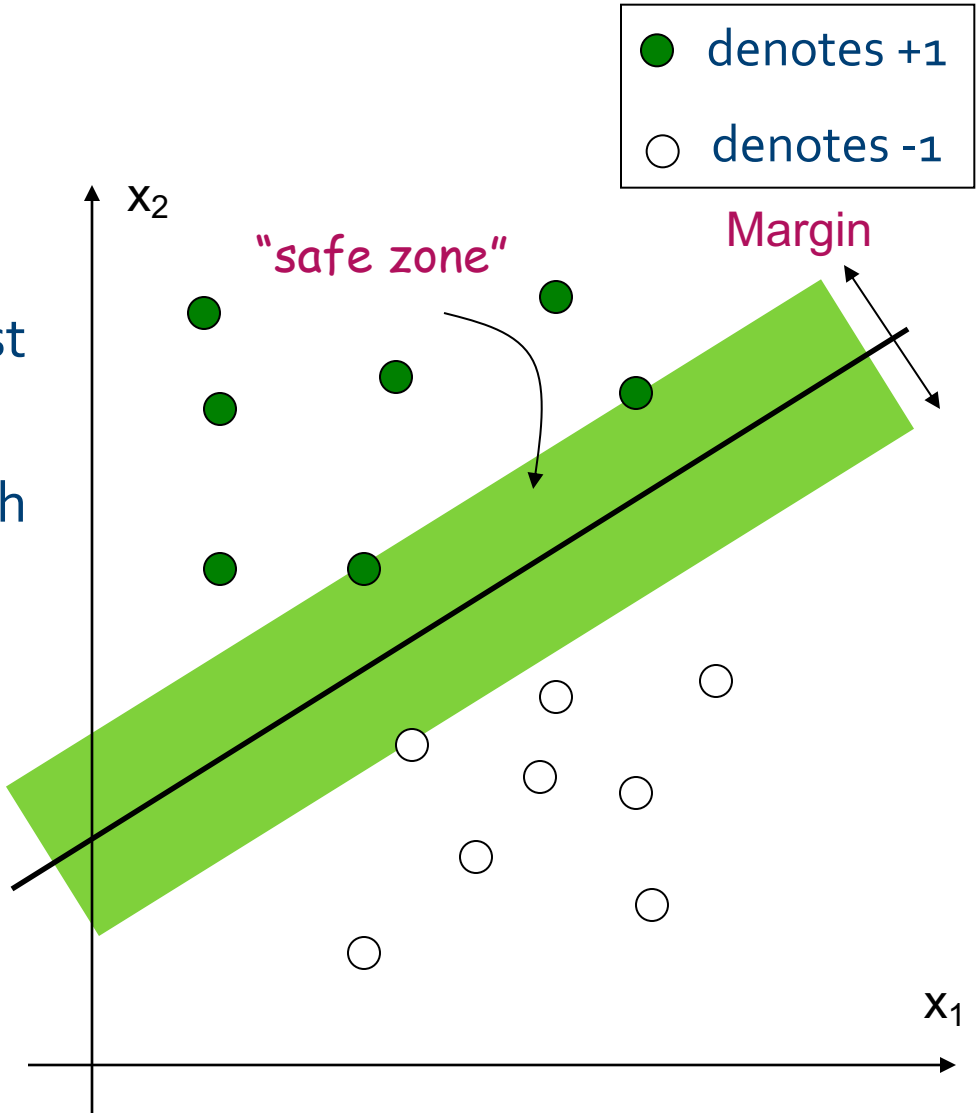


Large Margin Linear Classifier

- The linear discriminant function (classifier) with the **maximum margin** is the best

- Margin is defined as the width that the boundary could be increased by before hitting a data point

- Why is this working well?
  - Robust to outliners and thus strong generalization ability



Large Margin Linear Classifier

- Given a set of data points:

$\{(\mathbf{x}_i, y_i)\},\ i = 1,2,\ldots,n$, where

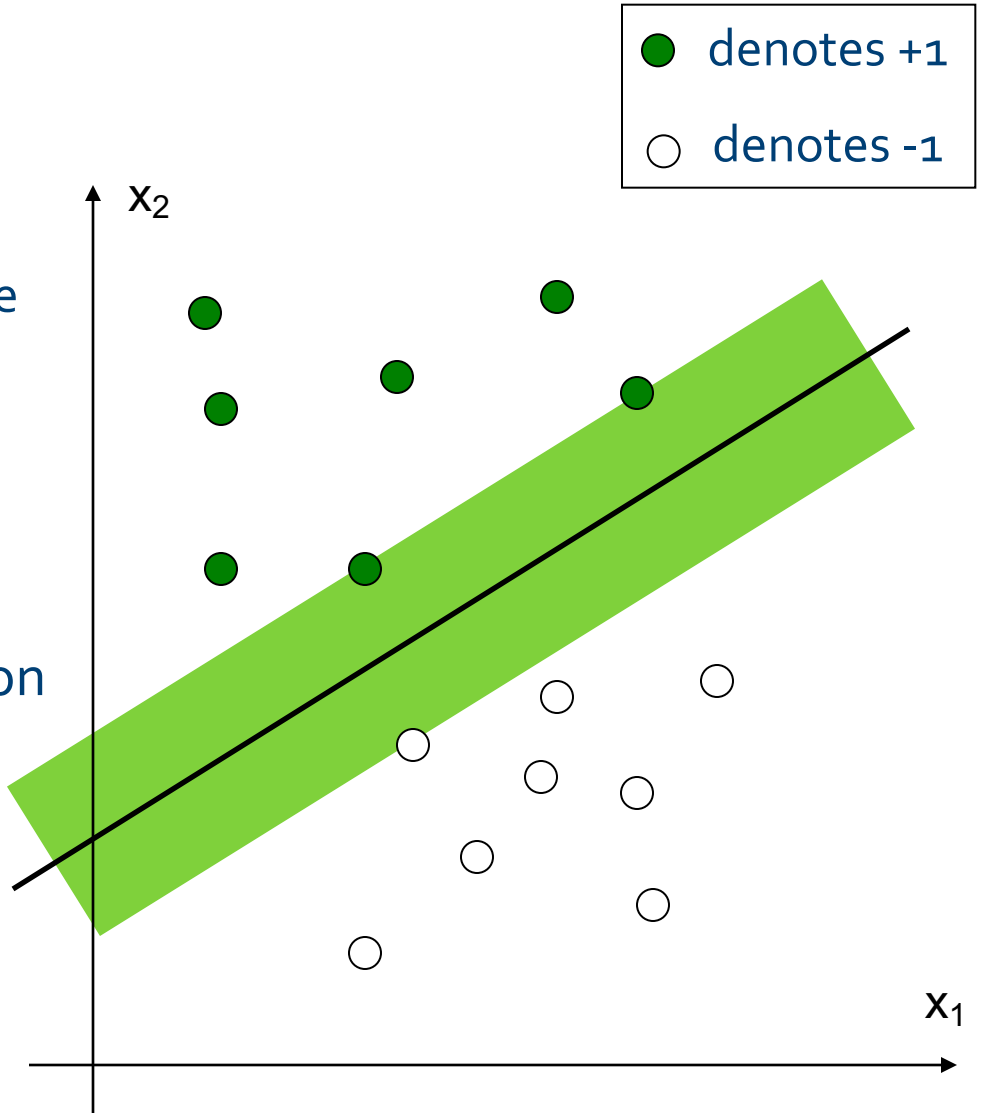
$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b > 0$$

$$\text{For } y_i = -1,\quad \mathbf{w}^T\mathbf{x}_i + b < 0$$

- With a scale transformation on both *w* and *b*, the above is equivalent to

$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1,\quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



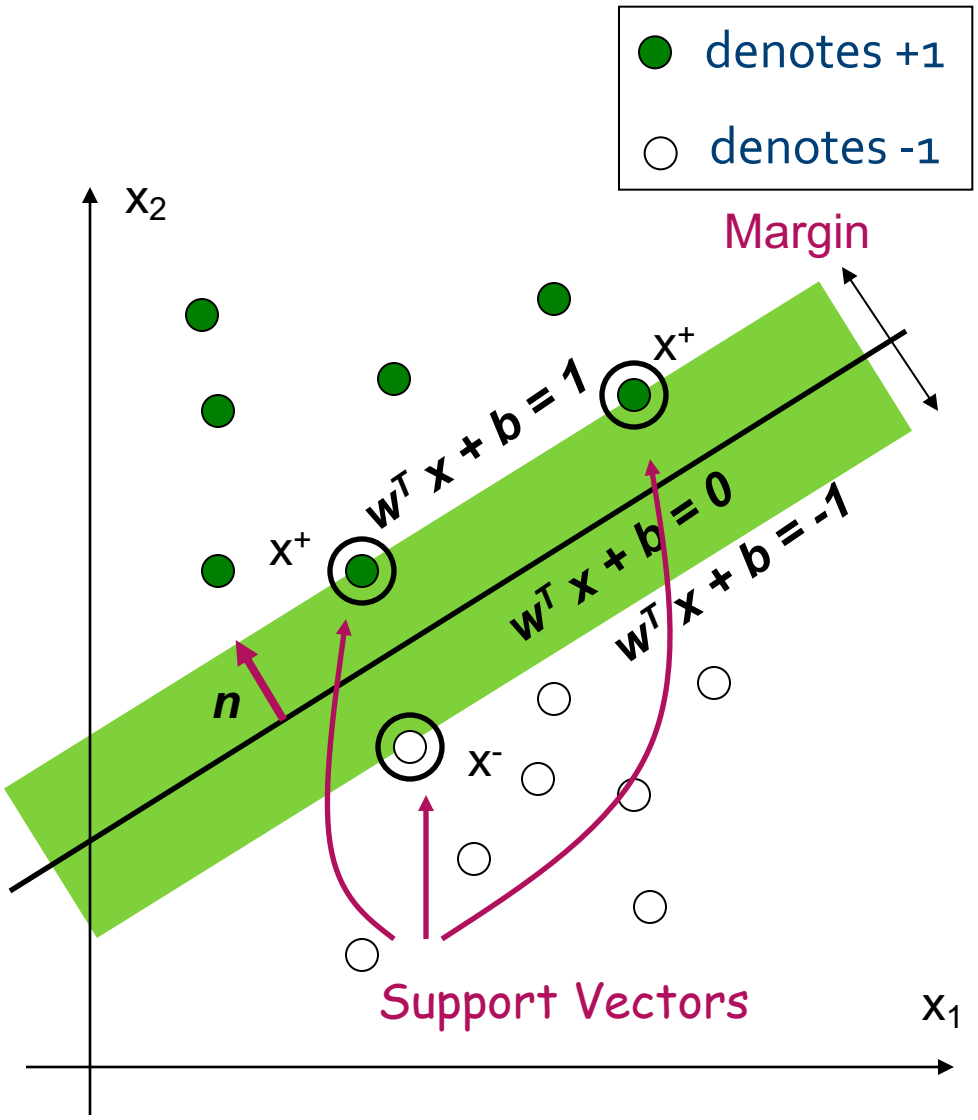
Large Margin Linear Classifier

- We know that

$$\mathbf{w}^T \mathbf{x}^+ + b = 1$$
$$\mathbf{w}^T \mathbf{x}^- + b = -1$$

- The margin width is:

$$M = (\mathbf{x}^+ - \mathbf{x}^-) \cdot \mathbf{n}$$
$$= (\mathbf{x}^+ - \mathbf{x}^-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$



Large Margin Linear Classifier

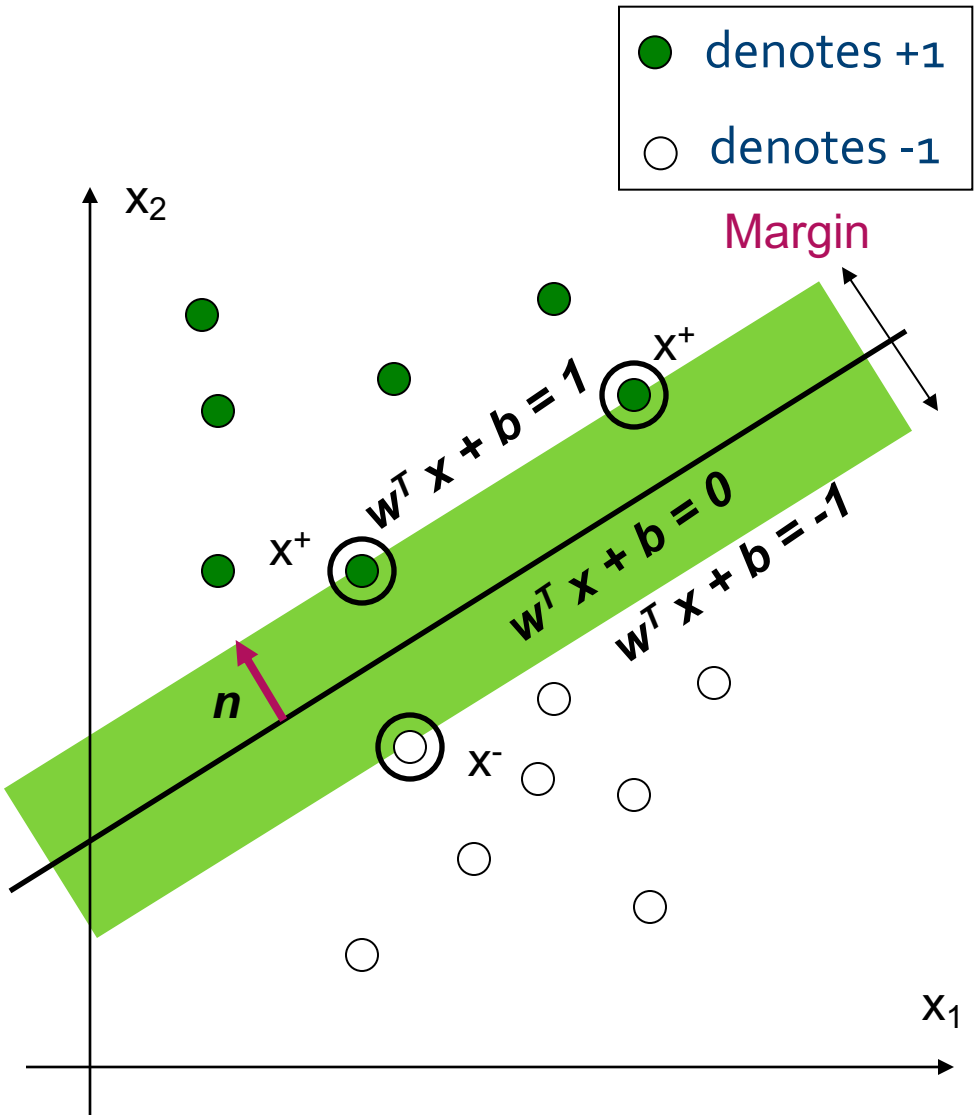
- Formulation:

$$\text{maximize} \quad \frac{2}{\|\mathbf{w}\|}$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$



Large Margin Linear Classifier

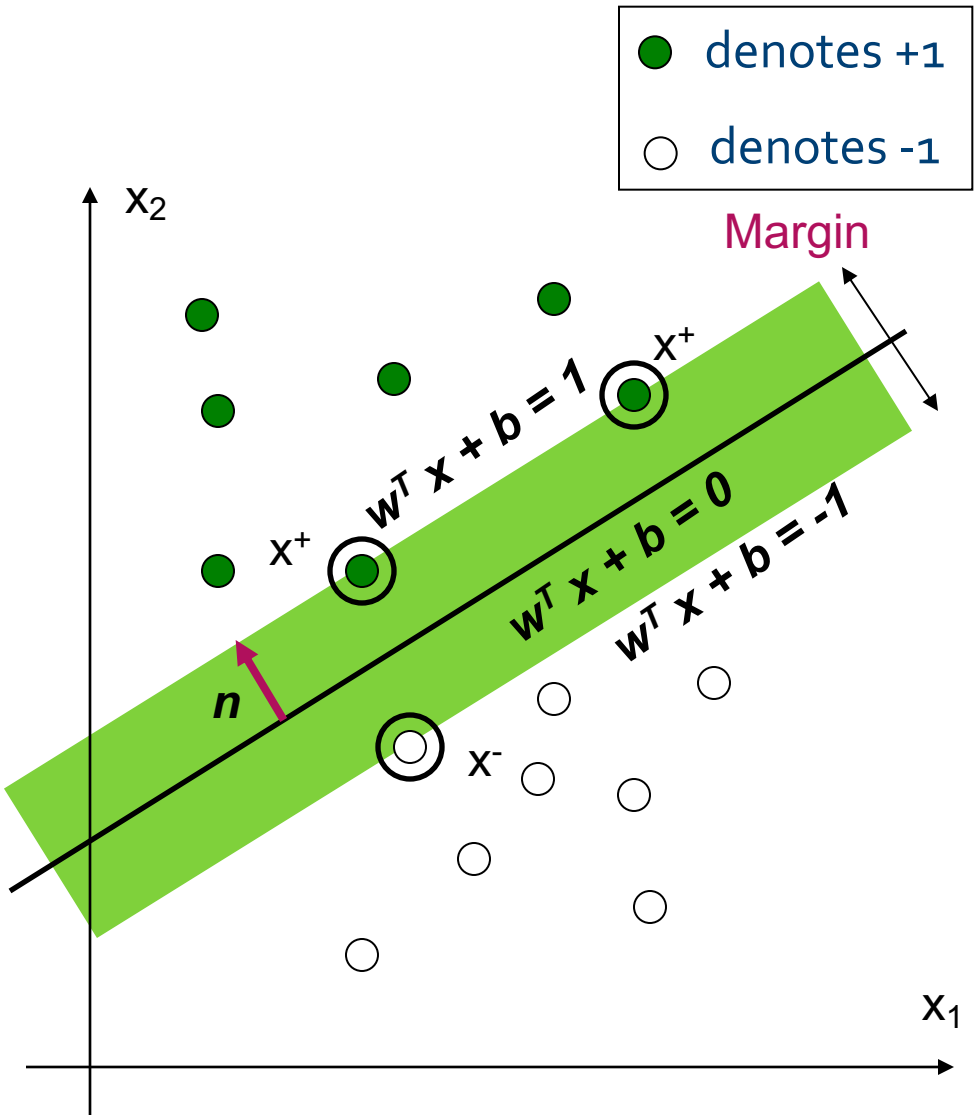
- Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



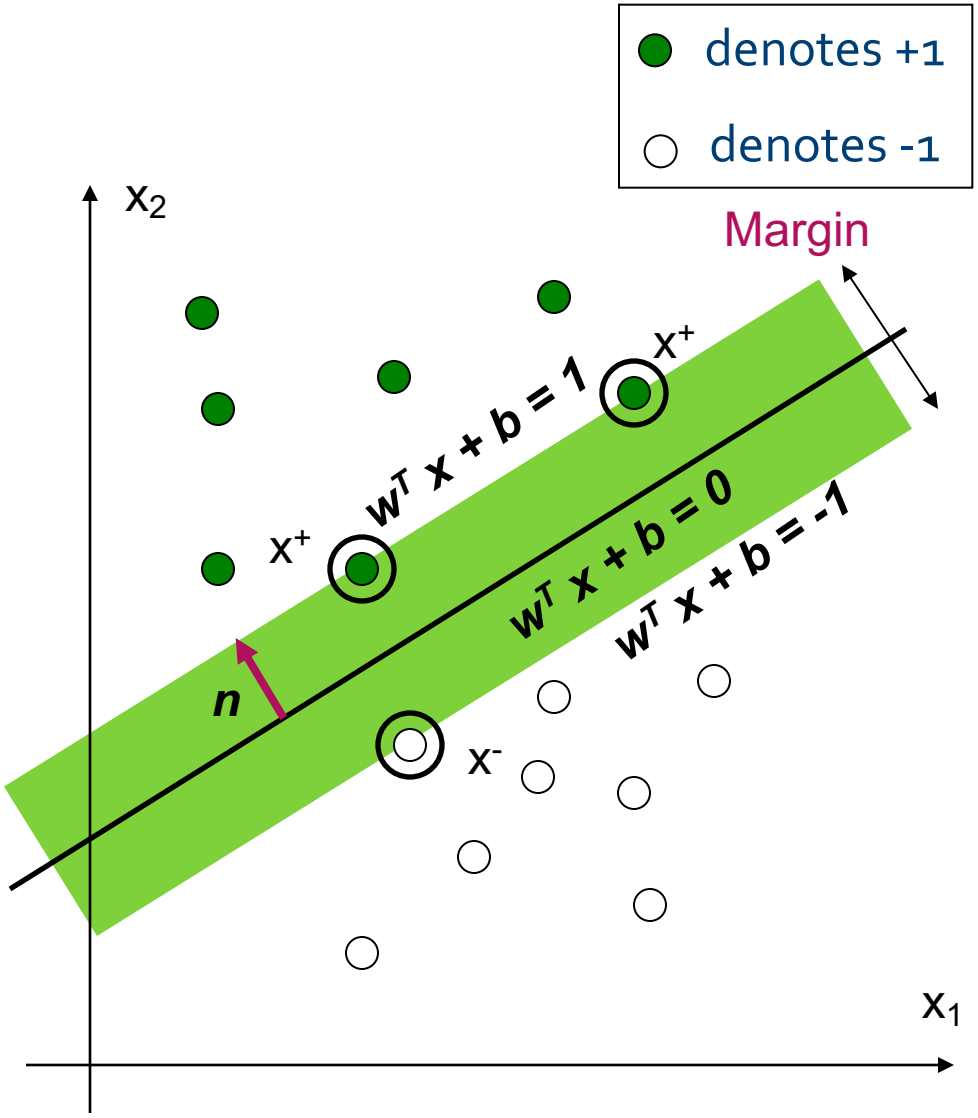
Large Margin Linear Classifier

- Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$



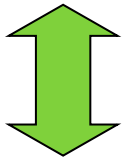
Solving the Optimization Problem

Quadratic programming with linear constraints

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

Lagrangian Multiplier Method



$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

Solving the Optimization Problem

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \qquad \alpha_i \geq 0$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

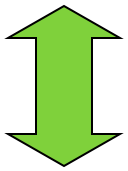
$$\frac{\partial L_p}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Solving the Optimization Problem

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

Lagrangian Dual Problem



$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{s.t.} \quad \alpha_i \geq 0 \quad \text{, and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Solving the Optimization Problem

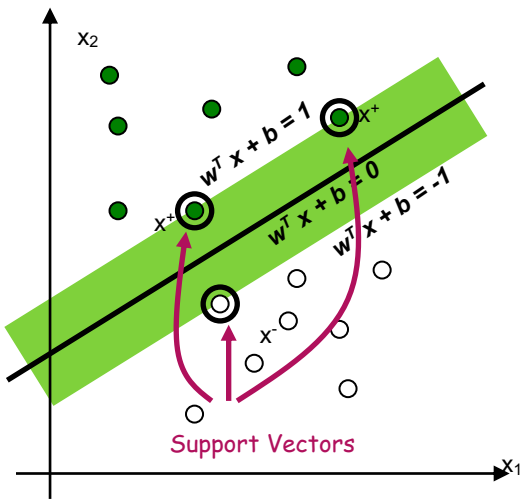
- From Karush–Kuhn–Tucker (KKT) conditions, we know:

$$\alpha_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) = 0$$

- Thus, only support vectors have $\alpha_i \neq 0$

- The solution has the form:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i \in \text{SV}} \alpha_i y_i \mathbf{x}_i$$



$$\text{get } b \text{ from } y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0,$$
$$\text{where } \mathbf{x}_i \text{ is support vector}$$

Solving the Optimization Problem

- The linear discriminant function is:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i \in \text{SV}} \alpha_i \mathbf{x}_i^T \mathbf{x} + b$$

- Relies on a **dot product** between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x_i}$

- Also keep in mind that solving the optimization problem involved computing the **dot products** $\boldsymbol{x_i^T x_j}$ between all pairs of training points

Empirical Studies with Text Categorization

- 10 Categories from Reuters-21578
- For a few categories, the SVM method significantly outperforms the KNN approach

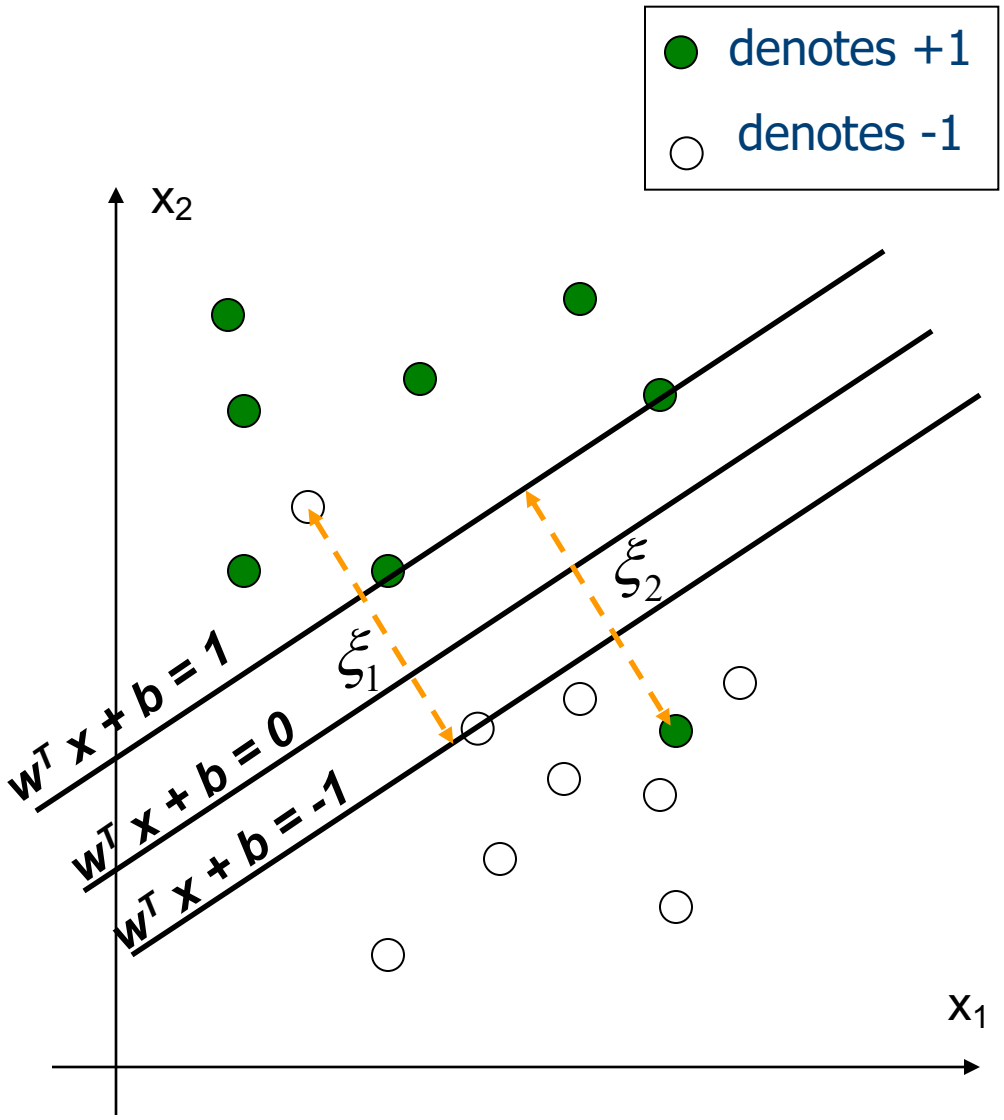
Category	KNN	SVM
earn	97.3	98.0
acq	92.0	93.6
money-fx	78.2	74.5
grain	**82.2**	**94.6**
crude	85.7	88.9
trade	77.4	75.9
interest	74.0	77.7
ship	**79.2**	**85.6**
wheat	**76.6**	**91.8**
corn	**77.9**	**90.3**

Classification accuracy

Large Margin Linear Classifier

- What if data is not linear separable? (noisy data, outliers, etc.)

- Slack variables ξ_i can be added to allow misclassification of difficult or noisy data points



Large Margin Linear Classifier

- Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- Parameter *C* can be viewed as a way to control over-fitting.

Large Margin Linear Classifier

- Formulation: (Lagrangian Dual Problem)

$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

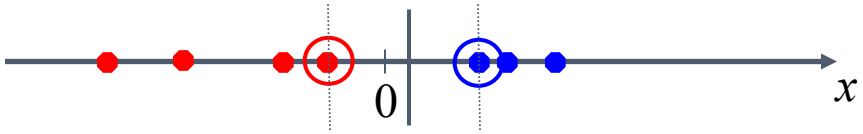
such that

$$0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

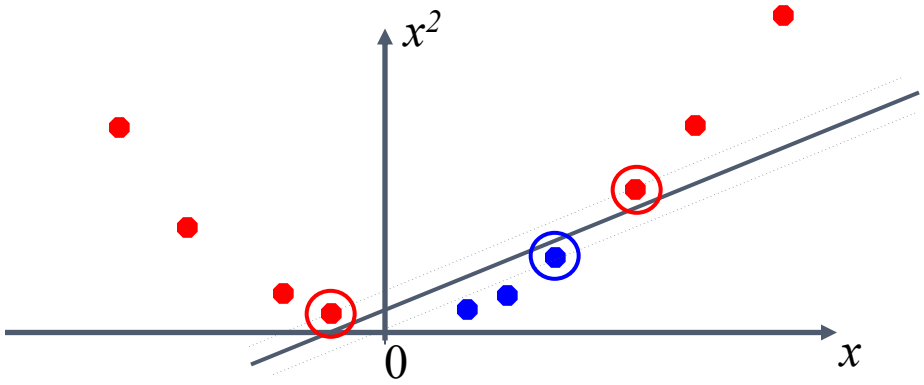
Non-linear SVMs

- Datasets that are linearly separable with noise work out great:



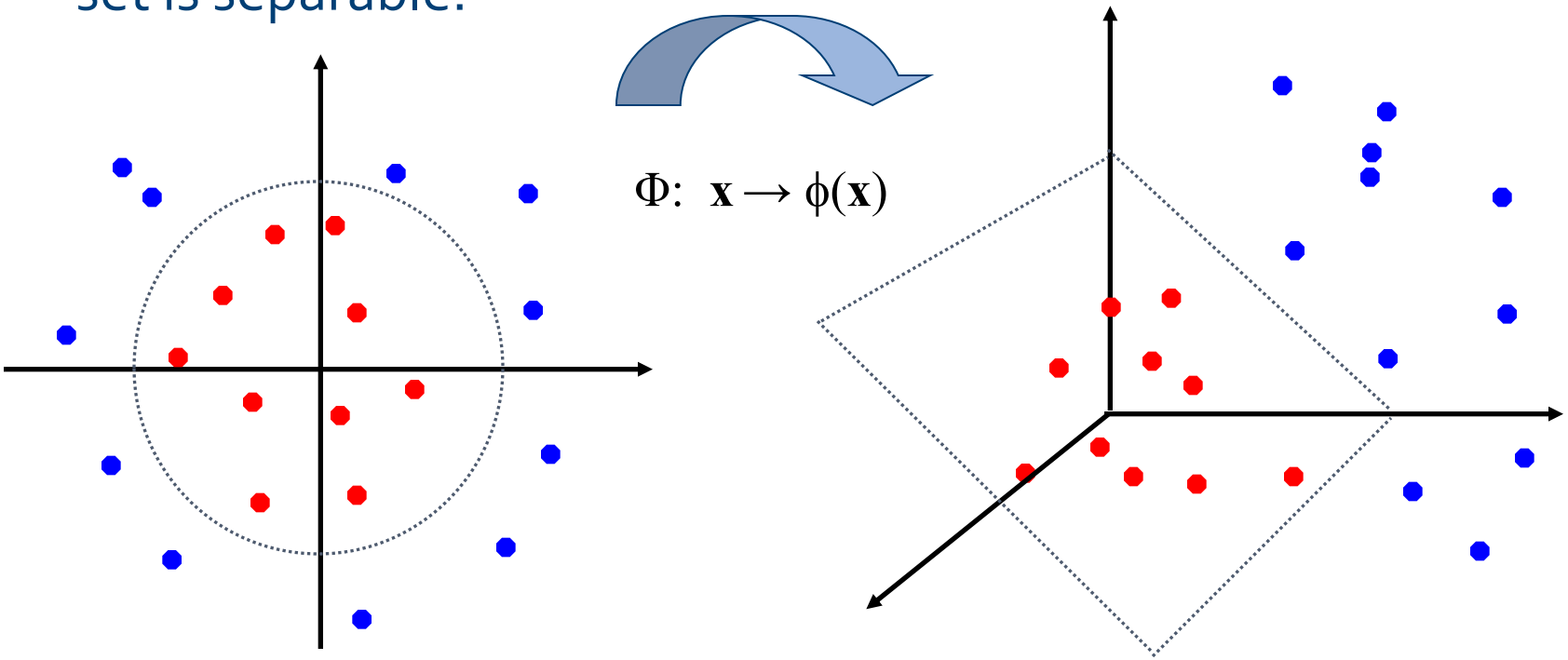
- But what are we going to do if the dataset is just too hard?

- How about… mapping data to a higher-dimensional space:

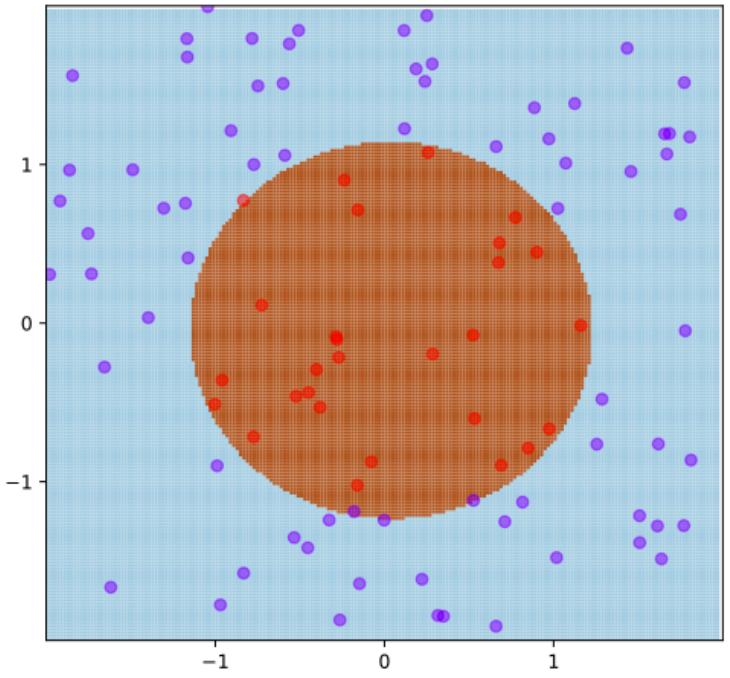


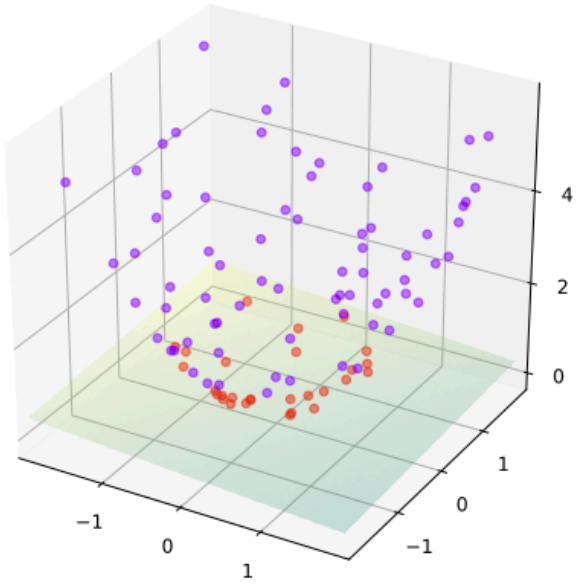
Non-linear SVMs: Feature Space

- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



The Kernel Trick

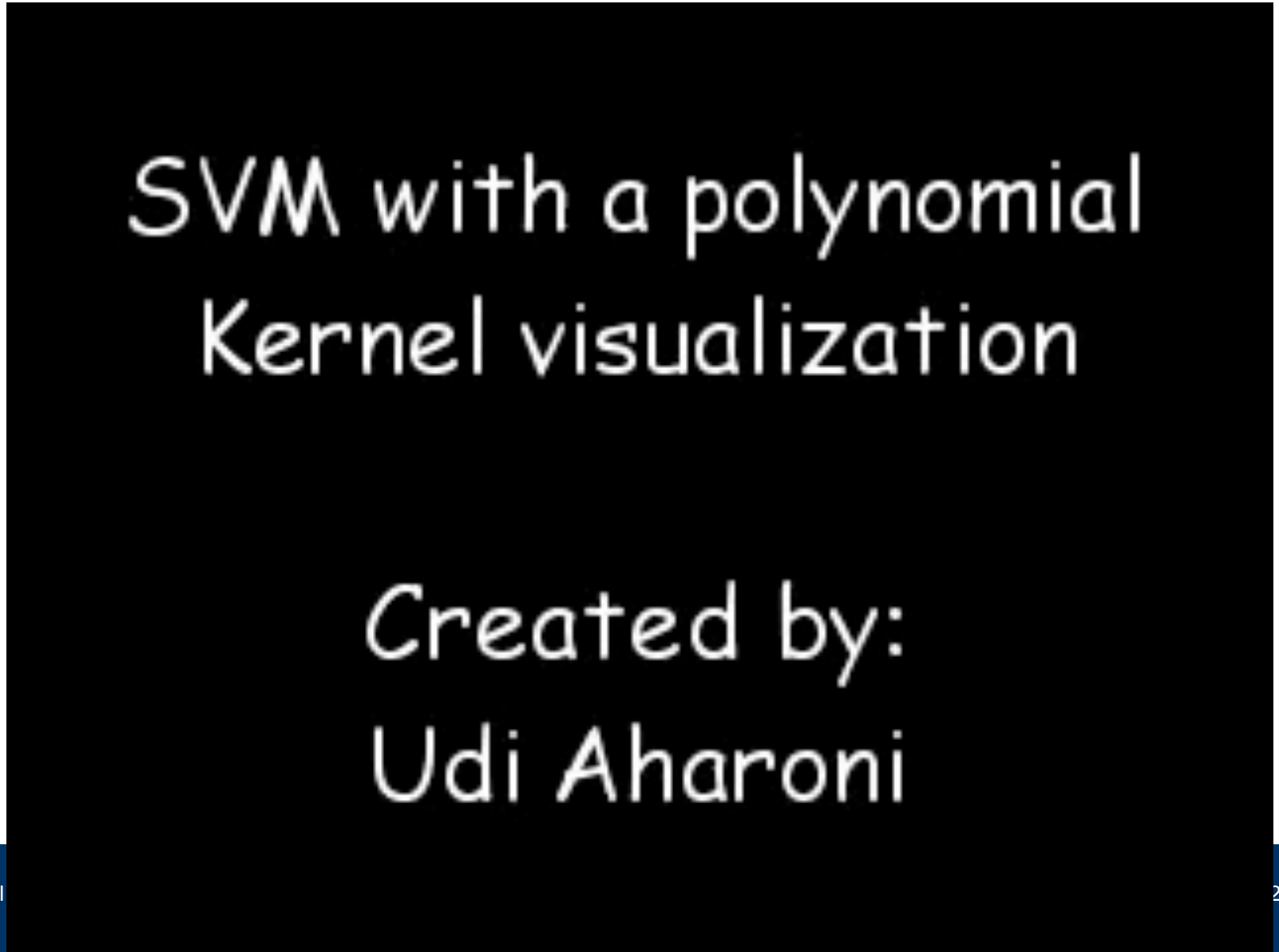




- SVM with kernel given by $\phi((a, b)) = (a, b, a^2, b^2)$ and thus $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \bullet \mathbf{y} + \mathbf{x}^2\, \mathbf{y}^2$. The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found.

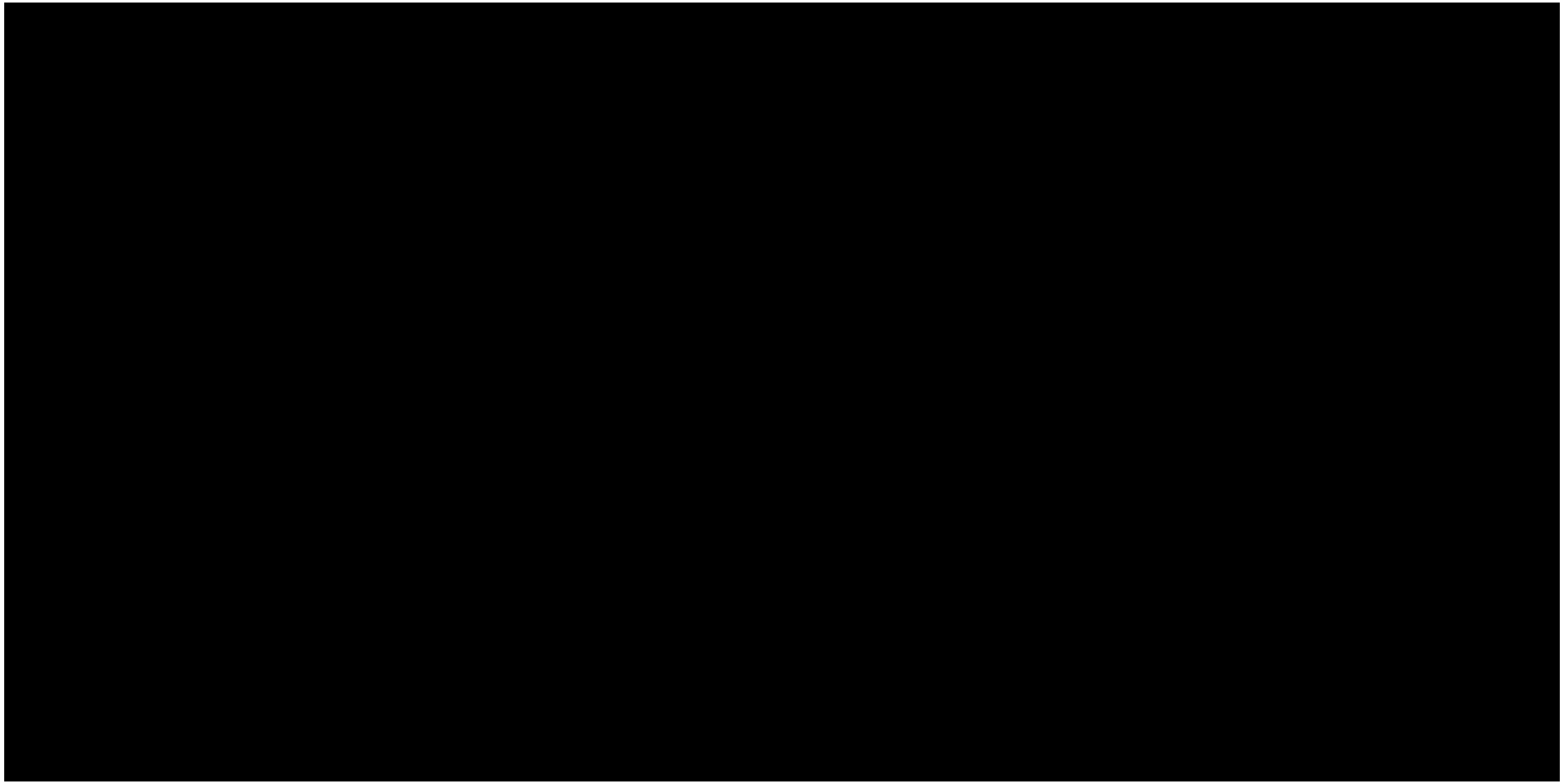
The Kernel Trick

https://www.youtube.com/watch?v=3liCbRZPrZA



The Kernel Trick

https://www.youtube.com/watch?v=9NrALgHFwTo



Nonlinear SVMs: The Kernel Trick

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \text{SV}} \alpha_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b$$

- No need to know this mapping explicitly, because we only use the **dot product** of feature vectors in both the training and test.
- A **kernel function** is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- Why use kernels?
  - Make non-separable problem separable
  - Map data into better representational space

Nonlinear SVMs: The Kernel Trick

- An example (to do):

2-dimensional vectors $\mathbf{x}_i=[x_{i1}\ \ x_{i2}]^T$;

Let $K(\mathbf{x}_i,\mathbf{x}_j)=(1+\mathbf{x}_i^T\mathbf{x}_j)^2$,

We need to show that $K(\mathbf{x}_i,\mathbf{x}_j)=\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$:

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1+\mathbf{x}_i^T\mathbf{x}_j)^2 = 1+x_{i1}^2x_{j1}^2+x_{i2}^2x_{j2}^2+2x_{i1}x_{j1}+2x_{i2}x_{j2}+2\,x_{i1}x_{j1}\,x_{i2}x_{j2}$$
$$= [1\ \ x_{i1}^2\ \ x_{i2}^2\ \ x_{i1}\sqrt{2}\ \ x_{i2}\sqrt{2}\ \ x_{i1}x_{i2}\sqrt{2}\,]\,[1\ \ x_{j1}^2\ \ x_{j2}^2\ \ x_{j1}\sqrt{2}\ \ x_{j2}\sqrt{2}\ \ x_{j1}x_{j2}\sqrt{2}\,]^T$$
$$= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$$

where $\phi(\mathbf{x}) = [1\ \ x_1^2\ \ x_2^2\ \ x_1\sqrt{2}\ \ x_2\sqrt{2}\ \ x_1x_2\sqrt{2}\,]^T$

Nonlinear SVMs: The Kernel Trick

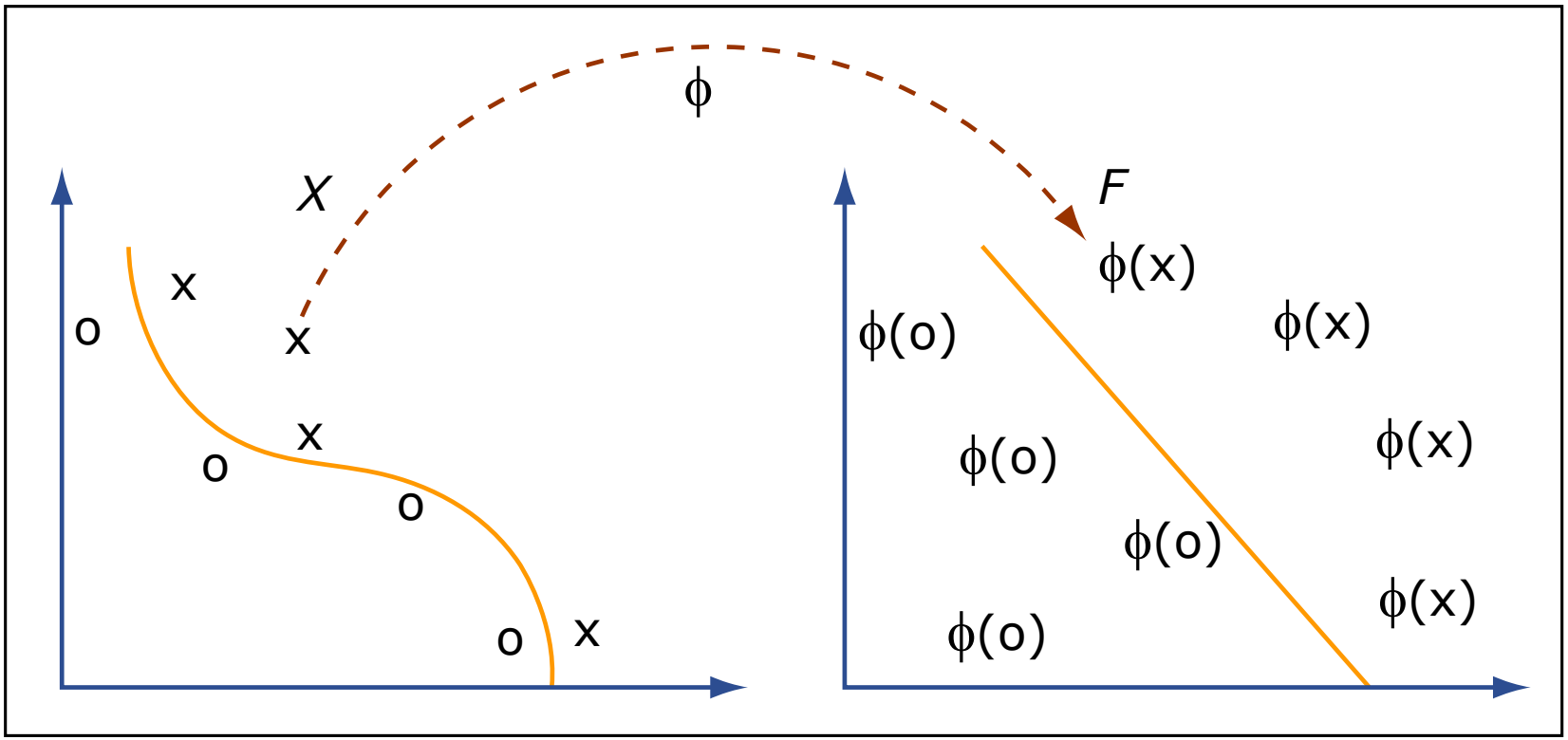


Image by MIT OpenCourseWare.

Nonlinear SVMs: The Kernel Trick

- Examples of commonly-used kernel functions:
  - Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
  - Gaussian (Radial-Basis Function (RBF)) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)$$

  - Sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$

- Generally, functions satisfying **Mercer's theorem** can be kernel functions:
  - Every semi-positive definite symmetric function is a kernel

Positive Definite Kernel

- Let X be a non-empty set. A positive definite kernel over $\mathbb{K}$ is a function $K:\aleph\times\aleph\to\mathbb{R}$ such that:
  - 1. K is symmetric.
  - 2. $\forall N\in\mathbb{N},\ \forall(x_1,x_2,...,x_N)\in\aleph^N,\ \forall(v_1,v_2,...,v_N)\in\mathbb{R}^N$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}v_i v_j K(x_i,x_j)\geq 0$$

Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem)

$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

such that

$$0 \le \alpha_i \le C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

- The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i \in \text{SV}} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- The optimization technique is the same.

Support Vector Machine: Algorithm

1. Choose a kernel function
2. Choose a value for C
3. Solve the quadratic programming problem (many software packages available)
4. Construct the discriminant function from the support vectors

Some Issues

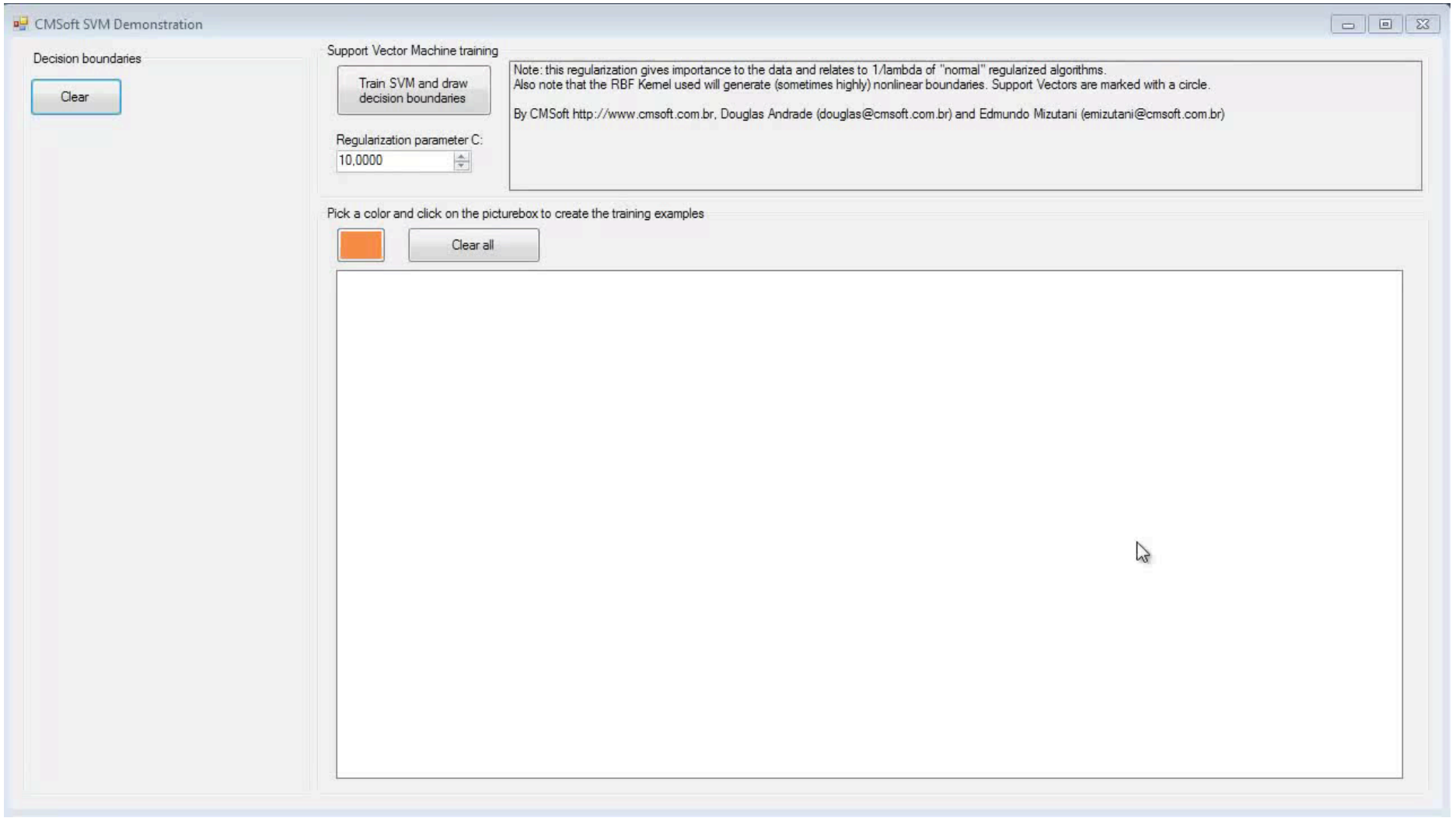
- Choice of kernel
  - Gaussian or polynomial kernel is default
  - If ineffective, more elaborate kernels are needed
  - Domain experts can give assistance in formulating appropriate similarity measures
- Choice of kernel parameters
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.
- Optimization criterion – Hard margin v.s. Soft margin
  - A lengthy series of experiments in which various parameters are tested

Summary: Support Vector Machine

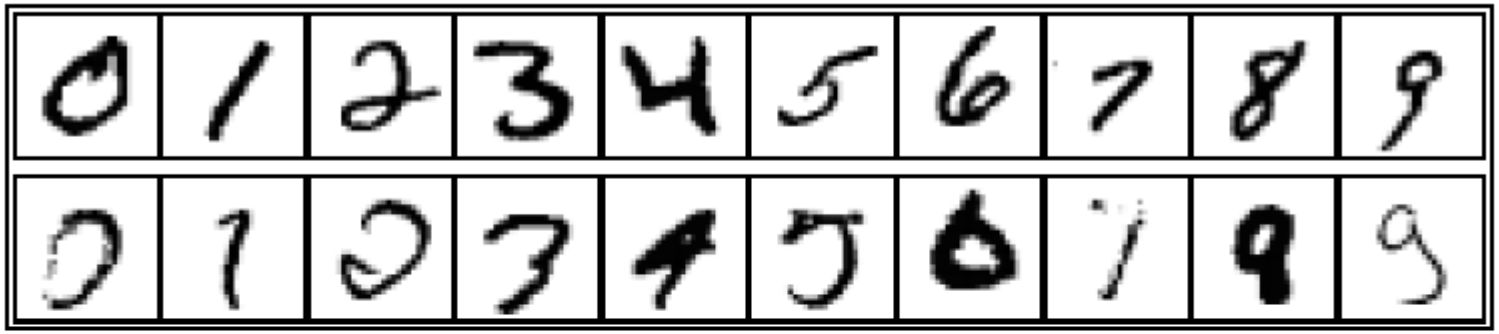
- 1. Large Margin Classifier
  - Better generalization ability & less over-fitting
- 2. The Kernel Trick
  - Map data points to higher dimensional space in order to make them linearly separable.
  - Since only dot product is used, we do not need to represent the mapping explicitly.

SVM demo

https://www.youtube.com/watch?v=bqwAlpumoPM



Handwritten digit recognition



3-nearest-neighbor = 2.4% error
400–300–10 unit MLP = 1.6% error
LeNet: 768–192–30–10 unit MLP = 0.9% error

Current best (kernel machines, vision algorithms) ≈ 0.6% error

- Not any more … but still one of the best ☺

Data and prior knowledge

SVMs

- Learning black-boxes
- Requires a large amount of high-quality training data

Real-world problems

- Data hard to obtain (cost, time, ethical reasons...)
- Seldom black-boxes: general and/or specific knowledge often available.

$\Longrightarrow$ need methods for the incorporation of PN into SVMs

Knowledge-Enhanced RBF kernel

In real-life problems, specific information relevant to the task is often available. A few examples:

- In climatology, measurements have known pseudo-periods: seasonal and diurnal (dominant frequencies).
- In anatomy, the weight of a specimen increases *w.r.t.* its dimensions and the increase is cubic (monotonicity, correlation patterns).
- In oncology, small and regular cells are typical while large and irregular cells are atypical (regions of the feature space).

KE-RBF kernels provide a way to leverage on such prior-knowledge.

ξRBF kernel

ξRBF kernel

$$K_a(x_1, x_2) = (\lambda + \mu\xi(x_1, x_2))K_{\text{rbf}}(x_1, x_2)$$

where $\xi : \mathcal{X}^2 \to \mathbb{R}$ contains the prior-knowledge and
$\mu = 1 - \lambda \in [0, 1]$ controls the the amount of prior-knowledge.

Motivation

Induce appropriate modifications to the kernel distance according to the prior-knowledge.

Types of prior-knowledge

- Unlabeled sets (similarity)
- Frequency decomposition

ξRBF kernel

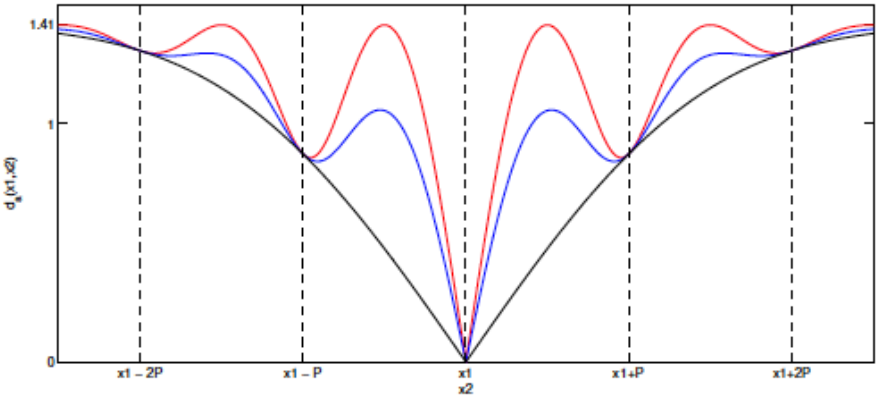
Frequency decomposition

$$K_a(x_1, x_2) = \left(\lambda + \mu \prod_{i=1}^{N_0} \xi_i(x_1, x_2)\right) K_{\text{rbf}}(x_1, x_2)$$

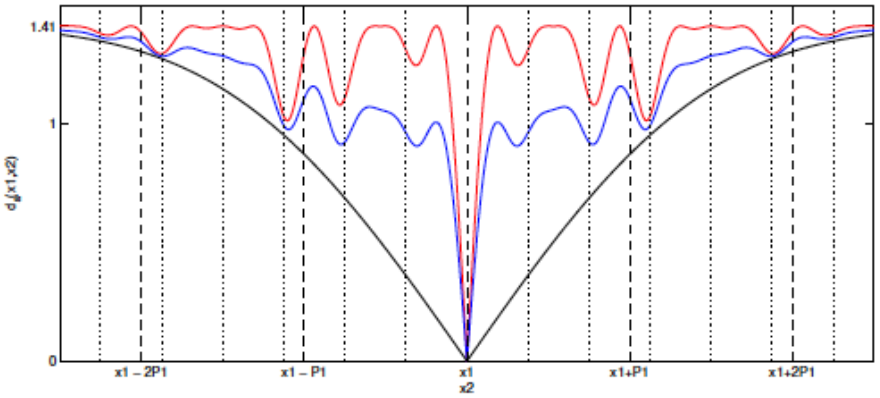
with

$$\xi_i(x_1, x_2) = \frac{\cos\left(\frac{2\pi}{P_i}(x_{1,j} - x_{2,j})\right) + 1}{2} = \frac{\cos(2\pi f_i(x_{1,j} - x_{2,j})) + 1}{2}$$

Single pseudo-period



Multiple frequencies



black $\mu = 0$, blue $\mu = 0.5$ and red $\mu = 1$

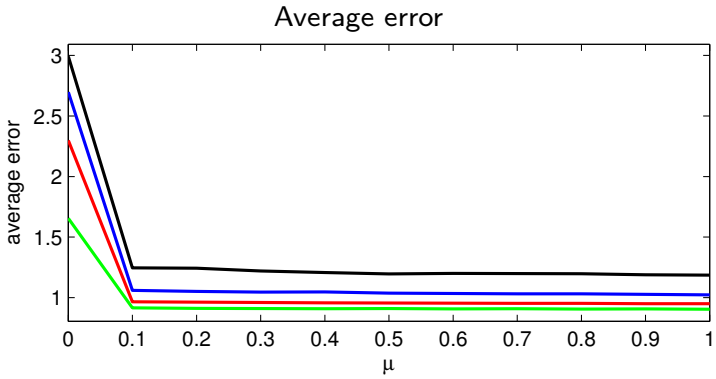
RBF kernel

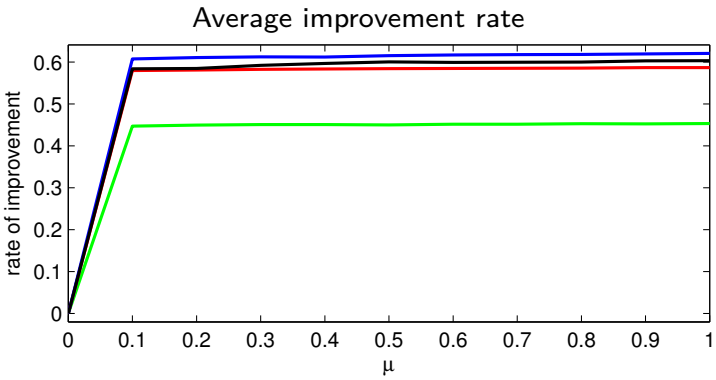
Application: meteorological predictions

- Prediction of daily temperatures in UK from 1914 to 2006.
- Publicly available from "UK Climate Projections" database.

Prior-knowledge

Cycle of seasons: pseudo-period of 365.25 days.



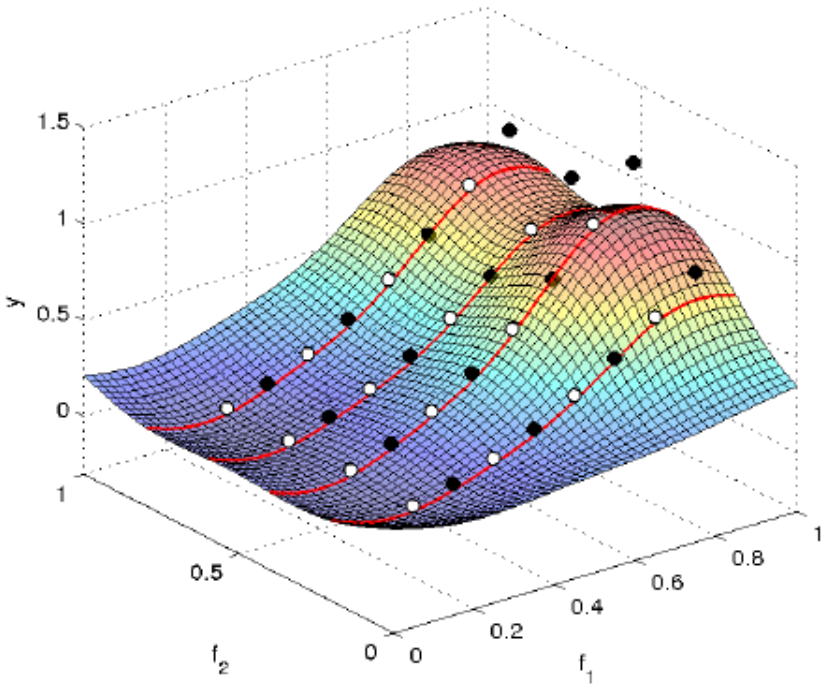


black $N = 50$, blue $N = 100$, red $N = 200$ and green $N = 400$

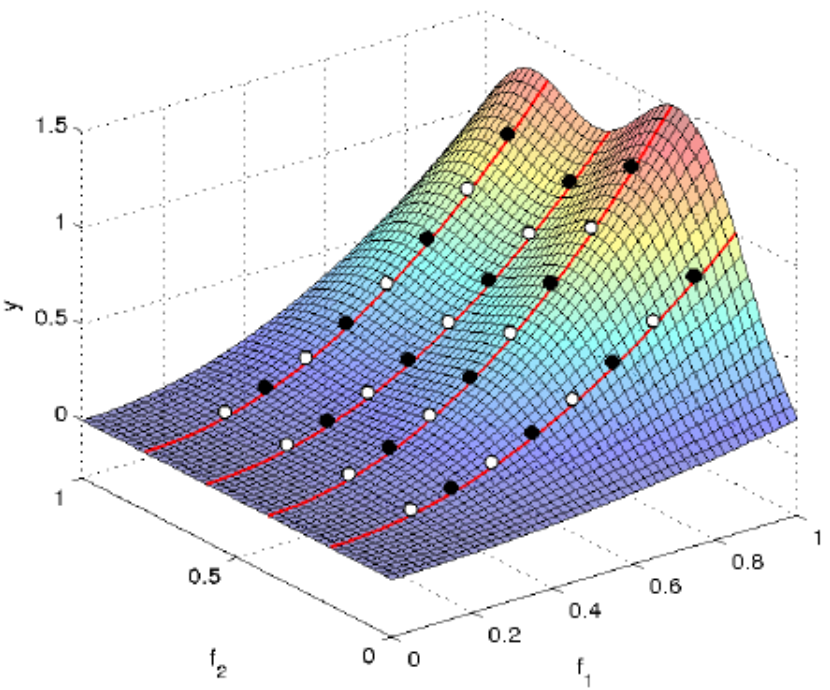
Missing data

Illustration: quadratic correlation.

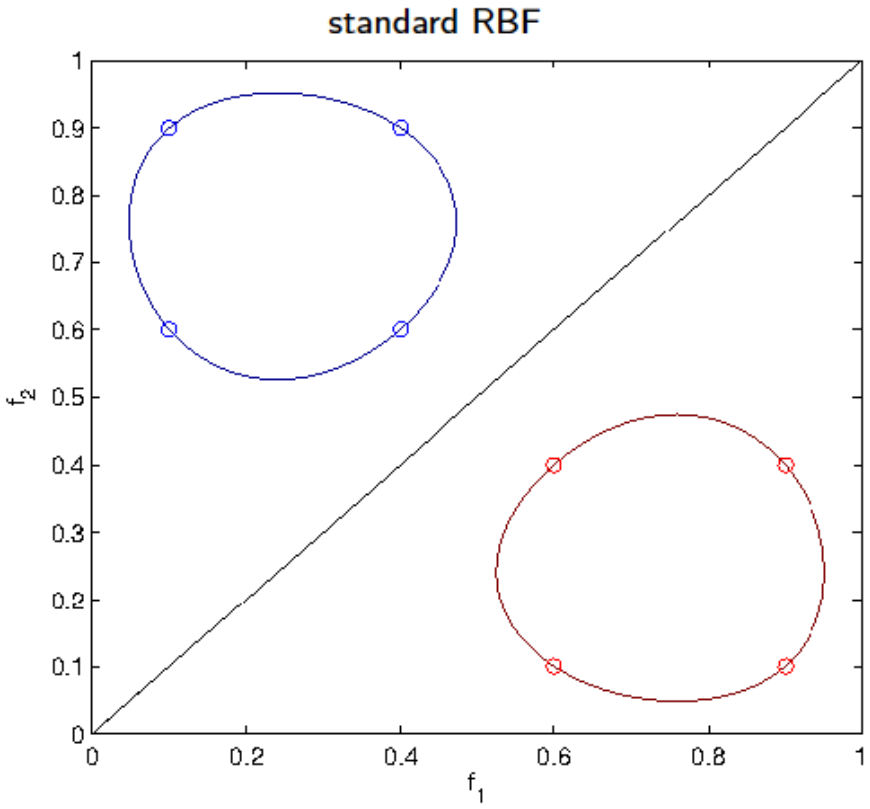
RBF

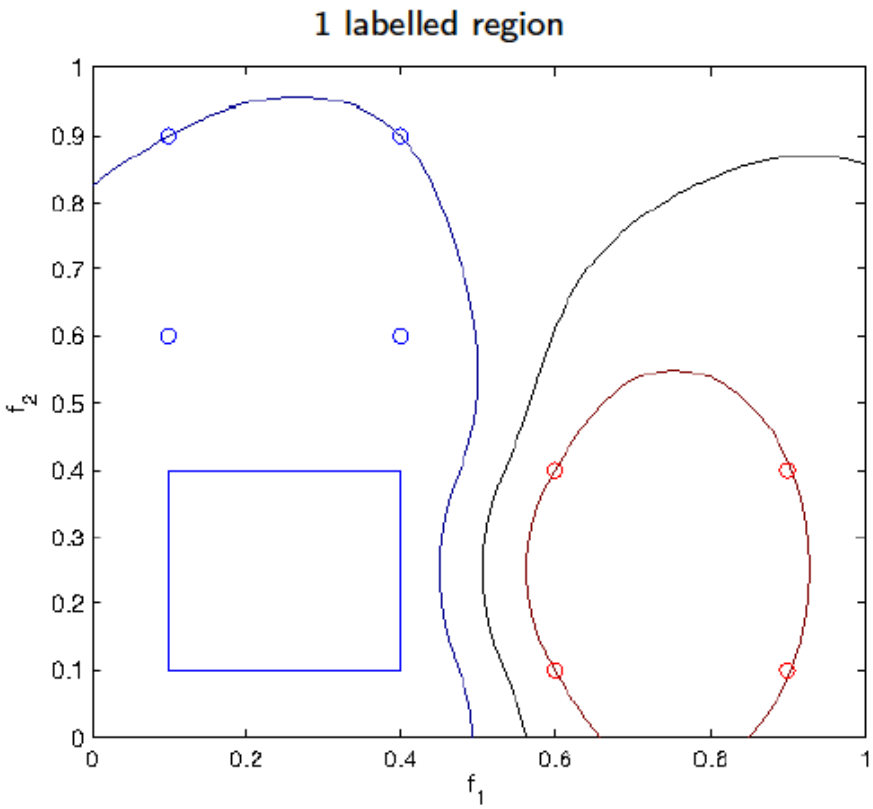


pRBF

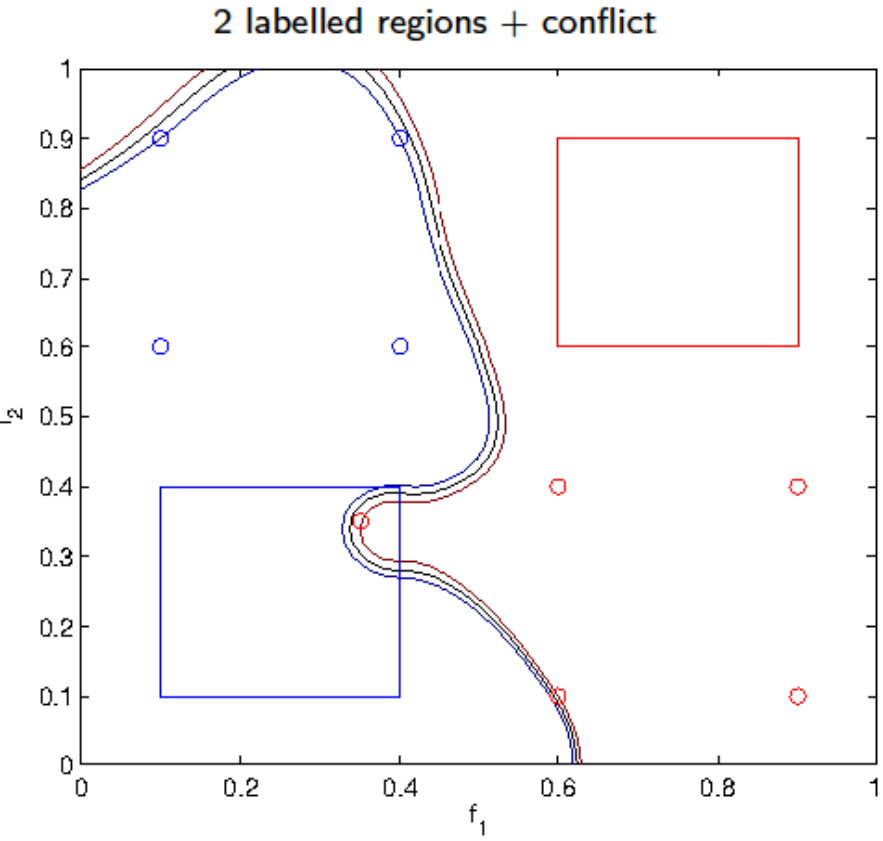


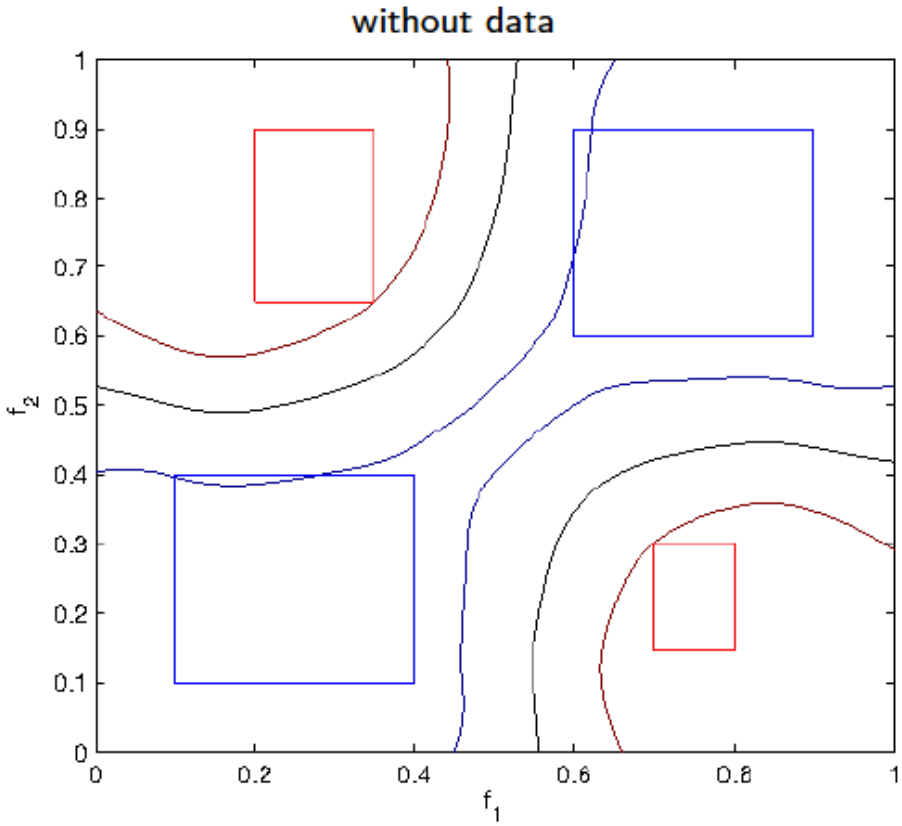
Examples (classification)



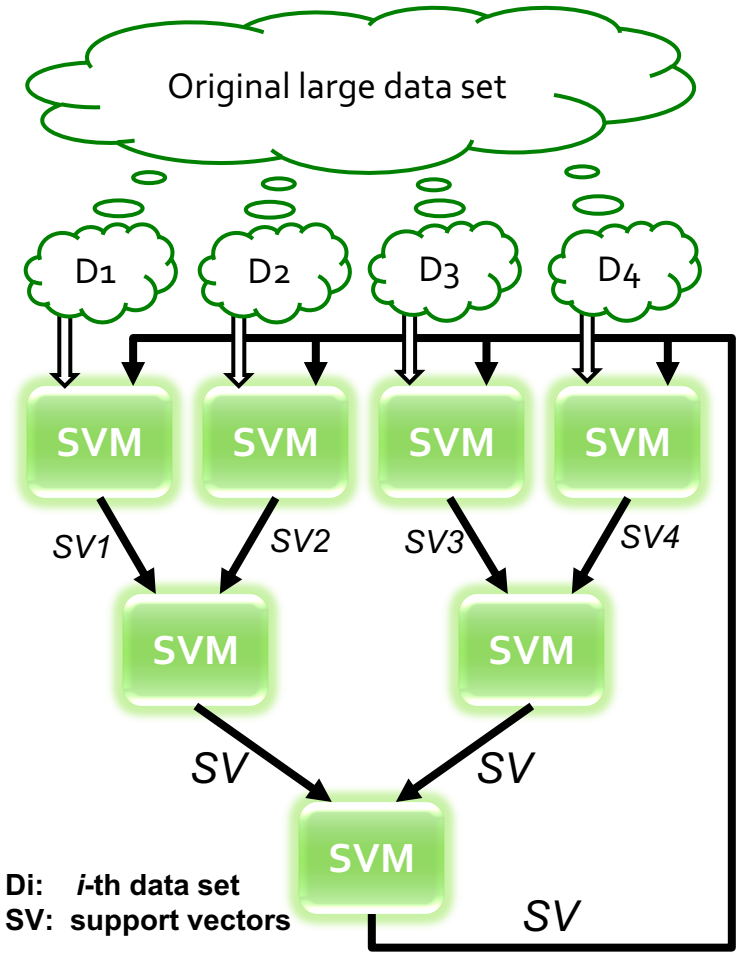


Examples (classification)





Cascade SVM



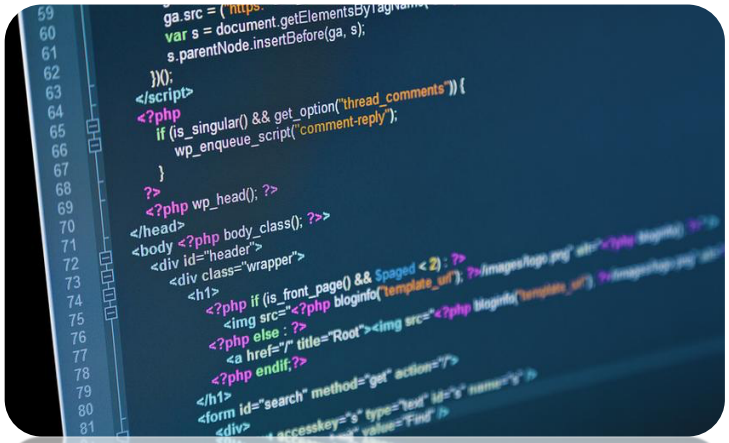
H. P. Graf, *Proc. Adv. Neural Inf. Process. Syst.*, 2004

- **Training process of basic SVM**
  - SVM training is time consuming:
    - Dominated by kernel evaluations;
    - $O(n^2)$ time complexity;
- **Parallel SVM (Cascade)**
  - Parallel processing of multiple smaller subsets
  - Partial results are combined in 2nd 3rd layer
    - workload in 2nd & 3rd layers is small.
- **Amdahl's law:**
  - Significant Speedup can be achieved if the runtime of the 1st layer dominates;
- **Global Convergence:**
  - Feed the 3rd layer result to 1st layer to check the KKT conditions.
    - The samples violating KKT conditions will join the next round of optimization.

Challenges

Machine Learning Applications on General-purpose CPU:

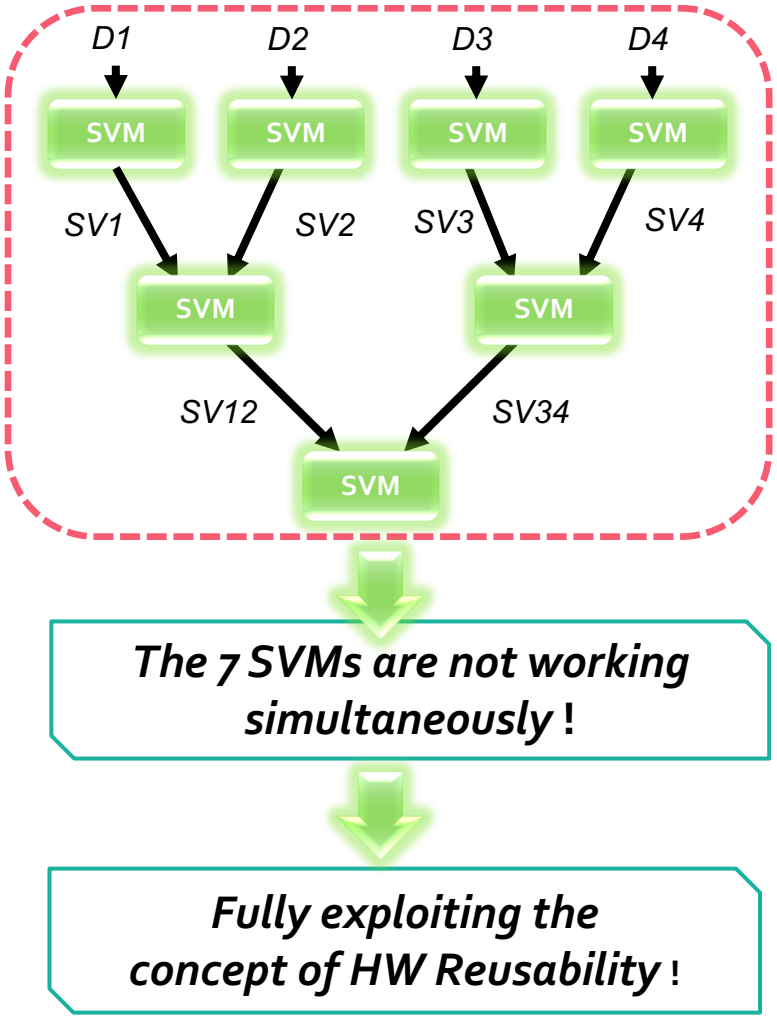
- Takes a huge amount of CPU time (e.g. several weeks or even months).
- Very high energy consumption.



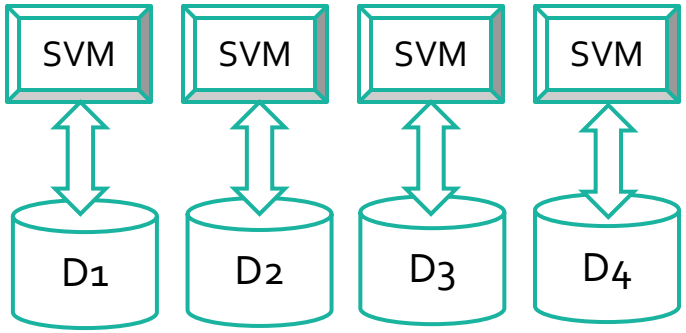
SOFTWARE SIMULATION

How to use moderate number of SVMs to construct HW architecture?

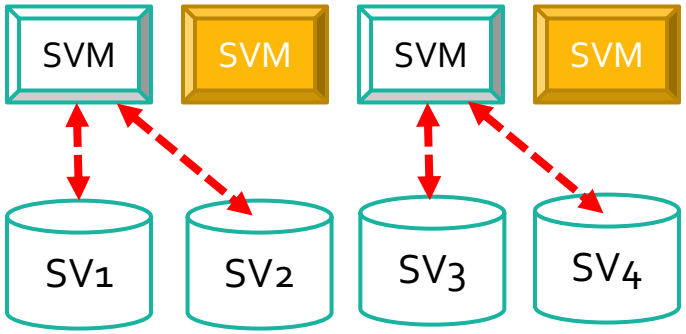
Software data flow of a Cascade SVM



- Implementation of 4 SVMs to perform 1st layer training:
  - D1~D4 stored in distributed memories.
  - SVMs access their private memories in parallel.

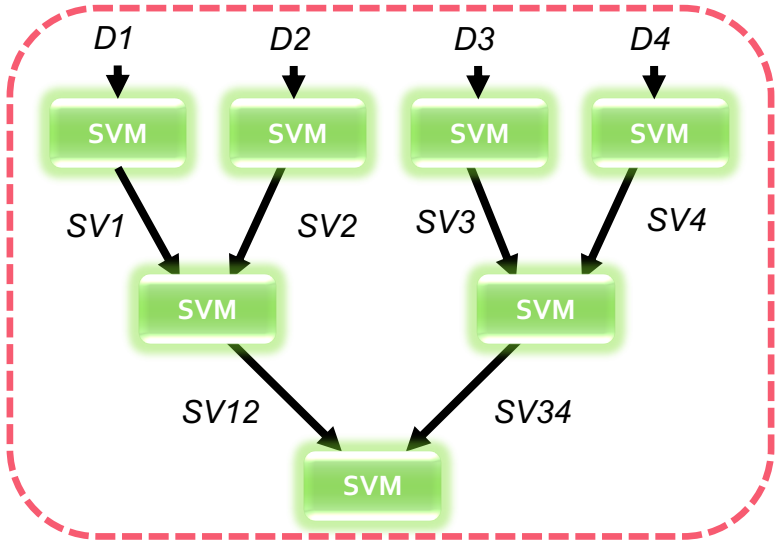


- For the 2nd layer, reuse 2 of the 4 SVMs. But how can they find *SV1* U *SV2* or *SV3* U *SV4*?
- Considering , simply need to enable each "reused SVM" access multiple memory blocks:



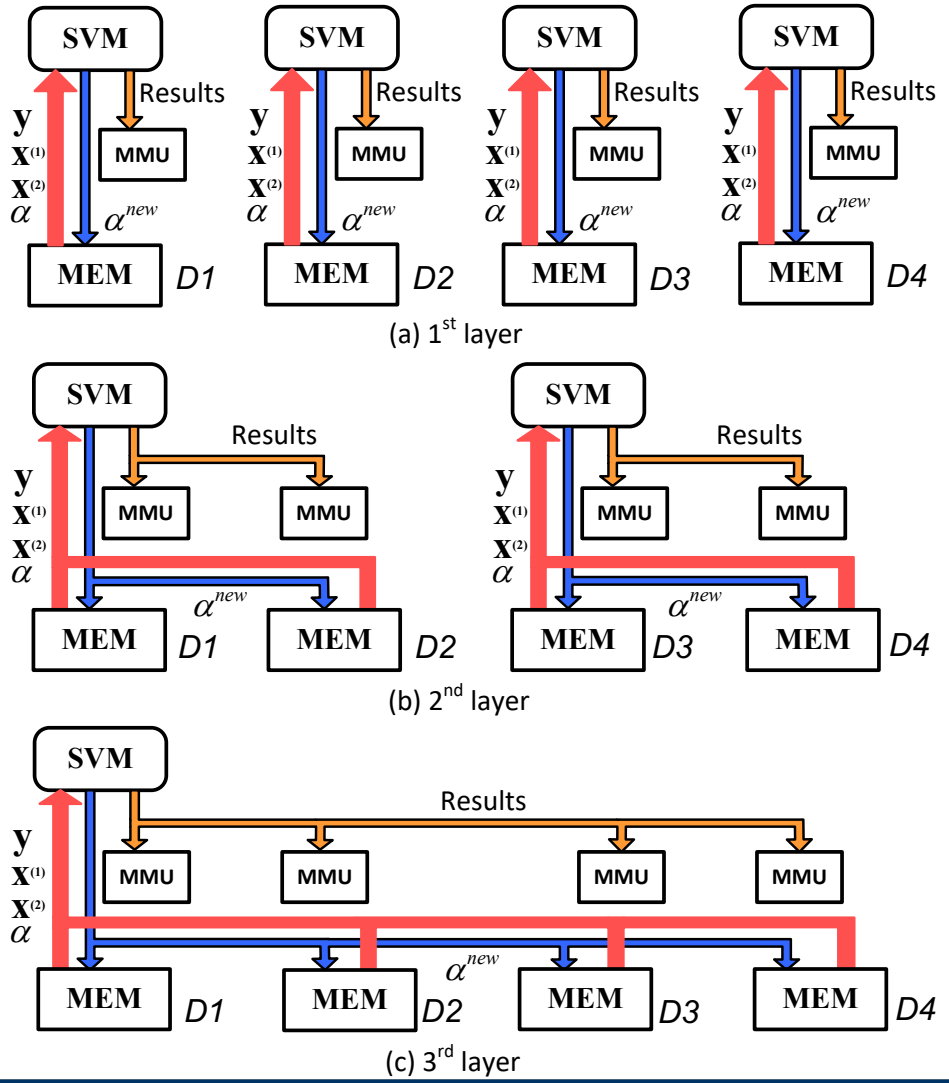
How to use moderate number of SVMs to construct HW architecture?

Software data flow of a Cascade SVM



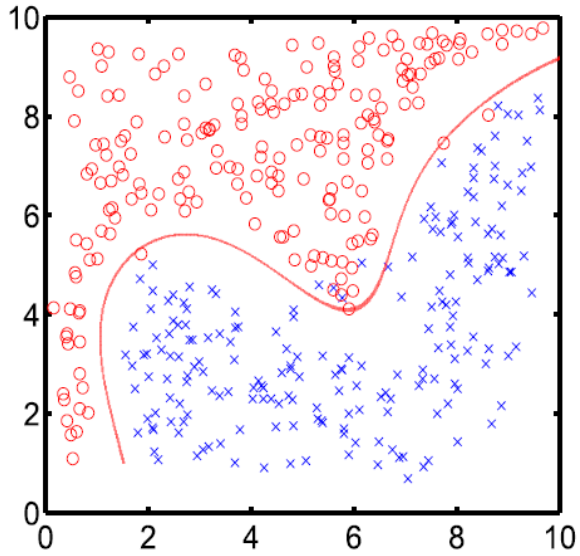
- *D1~D4* stored in *MEM1 ~ MEM4*;
- Implement 1st layer SVMs with HW, and reuse them for the following layers;
- Training results saved in MMU
- The final data flow is illustrated by the figure to the right:

Data flow of the HW architecture

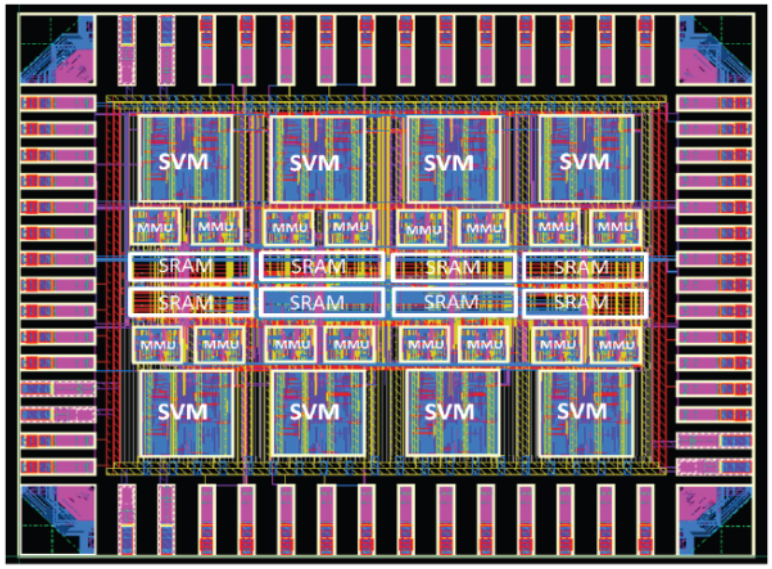


Experimental Results

- Synthesized using a commercial 90nm CMOS standard cell library;
- On-Chip memories generated by corresponding SRAM compiler;
- Layout generated using the same library, measure the area, power and maximum clock frequency (178MHz).



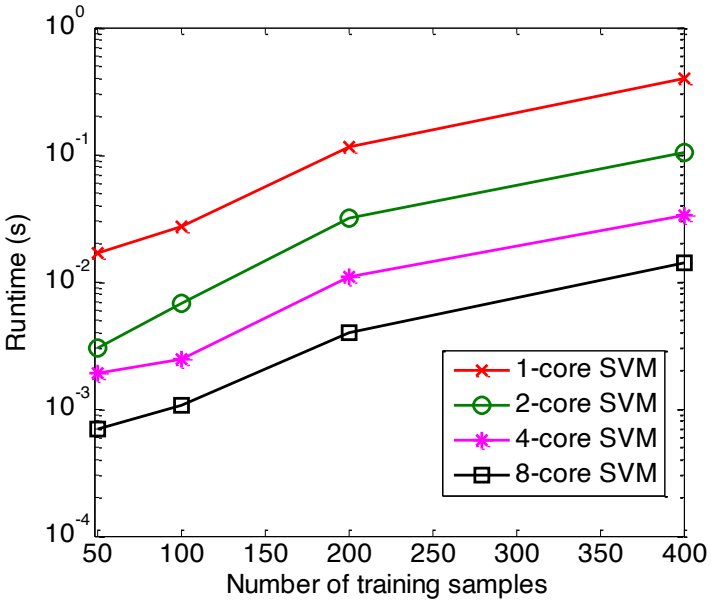
Decision boundary obtained from training 400 2-D samples.

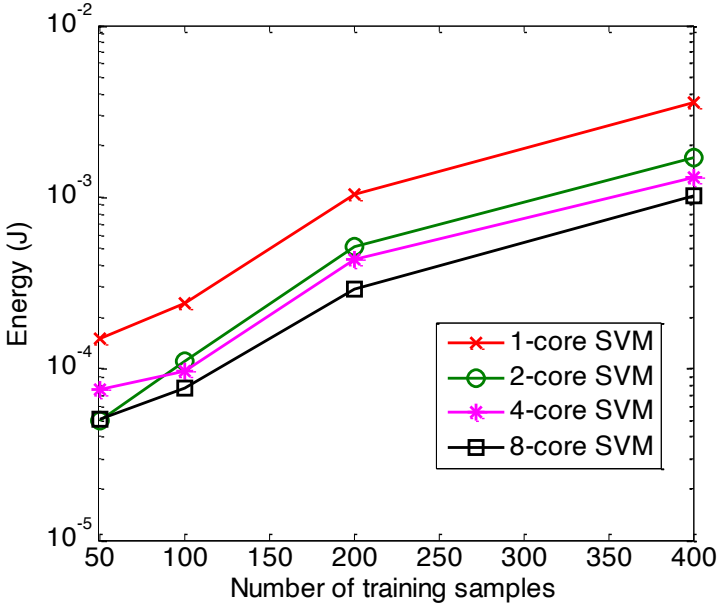


The 8-core design including I/O pads 6.68mm^2

Experimental Results

Datasets of different sizes to evaluate performance of each HW design





Focus on a fixed dataset

200 Samples	P (mW)	Area (um²)	Speed	Energy Reduction
Flat SVM	15.52	373,518	1x	1x
2-Core	27.74	727.946	3.67x	2.05x
4-Core	64.43	1,499,828	10.54x	2.54x
8-Core	126	3,143,700	28.79x	3.54x

- Energy = Runtime x Power

As number of cores increases:

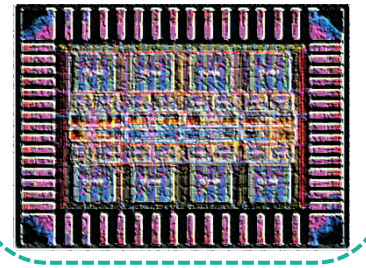
- Power & Area are "linearly" increased
- Speedup is increased much faster

Software Approach and Hardware Approach

C++ SVM program
Intel Pentium T4300
(2.1GHz) (45nm)

VS

ASIC designs of
Cascade SVMs
(178MHz) (90nm)



- Even the Intel CPU has a higher Clock frequency, and uses a more advanced technology, SVM ASIC designs can still outperform it by a lot!

Comparison of Runtimes and Energy Consumption